

June 1999

The USENIX Association Magazine

;login:

volume 24 • number 3

inside:

REPORTS FROM OSDI '99

SAGE NEWS & FEATURES:

Bailey Szeto on Sendmail Anti-Spam

How-To: Set Up an Apache Web Server

features:

The Tclsh Spot

by Clif Flynt

Source Code UNIX

by Bob Gray

and more . . .



USENIX

The Advanced Computing Systems Association

upcoming events

USENIX Annual Technical Conference

WHEN	WHERE	WHO
June 6-11/99	Monterey, CA	Avi Rubin, <i>Program Chair</i> Clem Cole & John Heidemann, <i>IT Coordinators</i> Jordan Hubbard, <i>Freenix Track Chair</i>

3rd USENIX Windows NT Symposium

WHEN	WHERE	WHO <i>program co-chairs</i>
July 12-14/99	Seattle, WA	Werner Vogels & Stephen Walli

2nd Large Installation System Administration of Windows NT Conference (LISA-NT)

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO <i>program co-chairs</i>
July 12-14/99	Seattle, WA	Gerald Carter & Ralph Loura

8th USENIX Security Symposium

WHEN	WHERE	WHO
August 23-26, 1999	Washington, D.C.	Win Treese, <i>Program Chair</i> Avi Rubin, <i>IT Coordinator</i>

2nd Conference on Domain-Specific Languages

Sponsored by USENIX in cooperation with ACM SIGPLAN and SIGSOFT

WHEN	WHERE	WHO <i>program chair</i>
October 3-6/99	Austin, TX	Thomas Ball

DEADLINES

Final Papers
August 24/99

2nd USENIX Symposium on Internet Technologies and Systems

Co-sponsored by the IEEE Computer Society Technical Committee on the Internet

WHEN	WHERE	WHO <i>program chair</i>
October 11-14/99	Boulder, CO	Fred Douglass

DEADLINES

Final Papers
August 31/99

3rd Annual Atlanta Linux Showcase

Co-sponsored by the Atlanta Linux Enthusiasts, USENIX, and Linux International

WHEN	WHERE
October 12-16/99	Atlanta, GA

13th Systems Administration Conference (LISA '99)

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO
November 7-12/99	Seattle, WA	David Parter, <i>Program Chair</i> Phil Scarr, <i>IT Coordinator</i> TBD, <i>Practicum Track Chair</i>

DEADLINES

Notification to Authors	Final Papers
June 30/99	September 1/99

Tcl/2K: The 7th USENIX Tcl/Tk Conference

WHEN	WHERE	WHO <i>conference co-chairs</i>
February 14-18/2000	Austin, TX	De Clarke & Tom Poindexter

DEADLINES

Paper Proposals	Notification to Authors	Final Papers
September 1/99	October 1/99	December 20/99

USENIX Annual Technical Conference

WHEN	WHERE	WHO
June 19-23/2000	San Diego, CA	Chris Small, <i>Program Chair</i> John Heidemann & John Kohl <i>IT Coordinators</i>

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
December 6/99	January 25/2000	April 25/2000

9th USENIX Security Symposium

WHEN	WHERE	WHO <i>conference chair</i>
August 14-17/2000	Denver, CO	TBD

4th Symposium on Operating Systems Design and Implementation (OSDI 2000)

WHEN	WHERE	WHO <i>conference co-chairs</i>
October 23-25/2000	San Diego, CA	Michael B. Jones & Frans Kaashoek

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
April 25/2000	June 29/2000	August 31/2000

contents

- 2 IN THIS ISSUE . . .
- 3 LETTERS TO THE EDITOR
- 7 THE USENIX CROSSWORD PUZZLE

CONFERENCE REPORTS

- 8 Reports on the 3rd Symposium on Operating Systems Design and Implementation (OSDI '99)

SAGE NEWS AND FEATURES

- 28 Where's the Horse?
by Tina Darmohray
- 29 Note from the President
by Hal Miller
- 30 Preparing to Consider Certification
by Bryan McDonald
- 31 SAGE Certification Update
by Gale Berkowitz
- 32 Applying sendmail Anti-Spam Checks at the User Level
by Bailey Szeto
- 43 Building a Linux Certification Program
by Dan York
- 46 How-To: Set Up an Apache Web Server
by Adam M. Donahue
- 53 Effective Perl Programming: A Perl CPANorama
by Joseph N. Hall

FEATURES

- 58 Musings
by Rik Farrow
- 60 Source Code UNIX: Security on a Source Code UNIX System
by Bob Gray
- 67 The Tclsh Spot
by Clif Flynt
- 71 Using Java: Remote Method Invocation
by Prithvi Rao

BOOK REVIEWS

- 76 The Bookworm
by Peter H. Salus

USENIX NEWS

- 78 USENIX Teams Up to Put on the 1999 Atlanta Linux Showcase
by Cynthia Deno
- 78 Incident Cost Analysis and Modeling Project II (I-CAMP II)
by Gale Berkowitz
- 79 20 Years Ago in USENIX
by Peter H. Salus
- 81 Call for Tutorial Proposals
by Daniel V. Klein
- 82 New Releases of *BSD and Debian Linux OSES Given Away at USENIX Annual Conference
by Cynthia Deno
- 82 USACO Wins Baltic Olympiad
by Rob Kolstad
- 83 Board Meeting Summary
by Ellie Young

ANNOUNCEMENTS AND CALLS

- 84 3rd USENIX Windows NT Symposium
- 86 LISA-NT – 2nd Large Installation System Administration of Windows NT Conference
- 89 3rd Annual Atlanta Linux Showcase
- 90 2000 USENIX Annual Technical Conference
- 96 **motd**
by Rob Kolstad



;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

\$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are \$80 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Editorial Staff

Editor:

Rob Kolstad <kolstad@usenix.org>

SAGE News and Features Editor:

Tina Darmohray <tmd@usenix.org>

Standards Report Editor:

Nick Stoughton <nick@usenix.org>

Managing Editor:

Jane-Ellen Long <jel@usenix.org>

Copy Editor:

Eileen Cohen

Proofreader:

Kay Keppler

Designer:

Vinje Design

Typesetter:

Festina Lente

Advertising

Cynthia Deno <cynthia@usenix.org>

Membership and Publications

USENIX Association

2560 Ninth Street, Suite 215

Berkeley, CA 94710

Phone: 510 528 8649

FAX: 510 548 5738

Email: <office@usenix.org>

WWW: <<http://www.usenix.org>>

©1999 USENIX Association. USENIX is a registered trademark of the USENIX Association. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this publication, and USENIX is aware of a trademark claim, the designations have been printed in caps or initial caps.

The closing dates for submission to the next two issues of *;login:* are August 3, 1999, and October 5, 1999.

in this issue . . .



by Jane-Ellen Long

Managing Editor

<jel@usenix.org>

As you read this, we're off to the land of sea otters and Steinbeck for the Annual Technical Conference, held this year in Monterey. If you're joining us, you'll enjoy dozens of exciting sessions, lots of great food, an invaluable chance to talk with people who share your interests, and a reception at the truly outstanding Monterey Bay Aquarium.

Can't make it this time? The scenery changes, but the excitement (and the great food) are S.O.P. for USENIX conferences. Check out our Upcoming Events on the inside front cover and the Announcements near the end of this issue, and remember to visit <<http://www.usenix.org/events/events.html>> regularly. Coming up next are Mount Rainier and Micro . . . , monuments and mustard mussels (the best appetizer in the world, IMHO, can be found in Georgetown, if you know where to look), and Silicon Hills and Highland Lakes. You might think of presenting a paper at one of our Y2K conferences – it's a great way to justify attending.

Another way to get to a conference is to offer to write Conference Reports for *;login:*. In this issue (I was bound to get to the point eventually) we have summaries of OSDI '99, held in San Diego; our next will include the highlights of the Workshop on Embedded Systems.

And more: See the great advice from Bailey Szeto on how you can let your users select their own anti-spam policies under sendmail. The SAGE How-To series continues with step-by-steps on setting up an Apache server. Other how-to highlights: how to profit from the dauntingly large CPAN archive of Perl modules, how to Tcl choice databites out of files, how to write Java applets that invoke methods on other machines. Don't skip the Letters column, where Open Source wars continue to rage. Or Musings, where Rik Farrow comes out of the (political) closet. Or the takes on certification: Tina Darmohray, Bryan MacDonald, and Dan York all offer plenty to think about. Or . . . but I'm keeping you from the good stuff, and I've got a date with a sea otter. See you next issue!

letters to the editor

OSS: Responses to Matthew Craighead

From Tim O'Reilly <tim@oreilly.com>:

Matthew, I wanted to urge you not to give up on Open Source yet! While it's true that there are a lot of people on slashdot (and around MIT, home of the FSF) who do in fact exhibit the values you describe in your letter, I don't think that this is true of the Open Source community in general. In fact, one of the whole reasons for the attempt to change the popular "meme" from Free Software to Open Source was to get away from some of the misconceptions you decry.

This has certainly led to some splits in the movement. For example, I am currently in the process of setting up a second "Open Source Summit" (to follow the Freeware Summit I organized last year), and I got heavily flamed by some people who thought I was out of line for inviting the Jini folks, even though their license isn't truly "free," as well as for suggesting an agenda that was looking to explore how much it is in fact licenses and how much it is network effects and other factors that account for the spread of popular open source programs.

But in the end, the people espousing the radical positions you decry in your letter are a small minority. Of the approximately 40 people coming to the summit (and these are about 50/50 split between developers of major Open Source software projects and people from corporations working with Open Source), only 2 or 3 were the source of the flames. And they ended up taking their marbles and going home, realizing their views were not shared by the vast majority of the attendees.

Despite the attempts of Richard Stallman to portray Open Source as a misguided attempt to recast the vision of free software, the Open Source movement is actually a recognition of the fact that a huge percentage of Open Source software developers don't care much about ideolo-

gy. They care about getting a better job done faster.

If you read Eric Raymond's paper "The Cathedral and the Bazaar" (<<http://www.ccil.org/~esr/writings>>), you'll see that it's really about the principles of distributed, community-based software engineering, not about free software ideology. (You might also want to look at my special issue of Esther Dyson's Release 1.0 newsletter, at <<http://www.edventure.com/release1/1198.html>>, which provides a big picture overview of Open Source.)

What's important about Open Source is that it's a recognition that what drives the success of Linux, FreeBSD, Perl, Apache, sendmail, and a host of other hugely successful Open Source software products is not ideology but science — software engineering methods and economic models suited to today's networked world.

From Con Zymaris <conz@cyber.com.au> :

I suspect you may have received a barrage of email on the publication of Matthew Craighead's letter in the recent *login*: [February 1999]. Here's my take at a response/refutation:

I believe that this letter should not have been published in a journal like *login*:. It belongs in the hodge-podge of idea-and-flame cauldron that is Slashdot. While I have no problems with the notion that Matthew Craighead has a right to his opinions, that doesn't mean they should get picked up and placed as prominently as they were in *login*:, as the sole letter published in the journal. Why? Because the ideas are half-baked, and no better than most of the ideas espoused on Slashdot that the author so fervently decries.

Let me point out a few:

1) "Most Linux fans dislike Microsoft." Likely to be true, but then most serious IT professionals not in the Microsoft-dominated realm also dislike Microsoft. So what?

2) The idea that OSS is a reincarnation of Communism is rubbish. If anything, the current system of either monopolistic or oligopolistic software monarchies is the equivalent to feudal and nondemocratic societies. Why? Because OSS is revitalising an otherwise tired and grey industry. It, through making available the source, is making dozens of startups spring forth as viable system, platform, and infrastructure software/service providers, all with a common, standardized, open-protocol base. This is real competition. As Bill Gates has famously stated, operating systems are a natural monopoly. If so, I want my operating system to not be controlled by one, single, all-powerful vendor. I want many OS source options. *This* is true competition. The current regime exudes an aura of "all edicts come from the centralized, Seattle-based politburo, for their own financial gain; take-it-or-leave-it." OSS is our (the users') way of "keeping the bastards honest."

3) There is a spread of individuals and groups involved in OSS. Very few are anti-commercial. What they are about is giving *users*, not corporations, more rights. Free software (I'll use the term interchangeably with OSS, I'm not religious about it) can and should be provided with *commercial* support, which users can then take or leave. Even the MIT-spawned FSF is perfectly happy with this.

4) Repeating what various anonymous people said on Slashdot is a pointless exercise. So what if someone said something crazy? Slashdot is a (marvelous) discussion platform for geeks, not the planning committee for OSS!

5) Obviously Matthew hasn't noticed that there are many companies and people making good money from supporting OSS. Cygnus, Red Hat, S.u.S.E., etc., are all profitable, and they all pay Linux developers with real money. There's no reason why a good Linux developer needs to be a "starving" Linux developer.

6) The point of the Halloween memos wasn't that Microsoft was unreasonably

more letters...

distressed by OSS; it is that they actually *were* distressed by OSS at all, which in itself is a major revelation. At that point, it looked like it had not even noticed Linux. Midway through last year, Microsoft was looking like it would have a perpetual reign as the IT industry's self-anointed Emperor; the thousand-year Reich. And along comes a band of "merry-men" (their words) who looked like they had a chance to unsettle this plan. Microsoft had analysed the threat, and found it to be *real*, and that none of their dirty tricks, FUD, breaking compatibility, proprietary protocols, etc., would work to defeat it. Did you actually read the memos? You will find that the analyst was talking about "breaking" open Internet protocols and the use of patents as their only chance of killing OSS off. Personally, I equate "breaking" Internet protocols as verging on criminal, and here was Microsoft planning this strategy as a fairly humdrum action. The selfishness and gall of this concept defies description. You wonder why people don't like them.

7) Linus Torvalds gets the best of all worlds. He's adored by the professional/technical IT community worldwide, gets to work on writing technically challenging free software OS kernels (his passion), and yet gets paid handsomely by Transmeta to do this and other work. He is *not*, however, in a position to kick himself for not having made millions off Linux. He himself has stated that the Linux kernel makes up only 1% of the Linux OS, and of this, only 5% is his work. Linux would never had gotten off the launch pad if it had not been developed and released as GPL OSS. The work is the blood, sweat, and tears of a million people worldwide, not one man.

8) OSS advocates do not want to make intellectual property illegal. Almost all, however, want to make software patents illegal. Many far more eloquent observers than I have written on the evils of software patents. Read them.

9) While it is true that Red Hat et al. have a business model that means that they don't have to create Linux from scratch, that is one of the tenets and strengths of OSS, not a weakness. Remember, a good programmer writes good code; a great programmer copies great code, and doesn't reinvent the wheel. This is a primary reason why OSS has mushroomed and developed great systems and apps quickly. Further, Red Hat, S.u.S.E., and Cygnus have all done their part in bringing out new code and useful extensions to Linux and GNU development tools, such as easier installation, GNOME, X-Free video drivers etc., all released as OSS.

10) You assume that people only develop software for monetary gain. This is an incorrect assumption. While OSS (in its extreme) may not attract the types of people who primarily focus on money, it will attract others who do it for reasons like contributing back to the OSS community, altruists, the talented and curious. Further, your comments are based on the premise that "closed-source" vendors actually make profit from their non-OSS software. This is mostly false. The advantages of OSS are that you can leverage existing code, produce a useful tool quickly, get it out the door for others to solidify and extend, thus cutting out 80% of the effort/cost required to create closed-source "commercial" software. Included in this cost are debugging and beta-test programs, paying for sales/admin/support/marketing staff, paying for marketing/advertising, paying for packaging, paying for printing of manuals, etc. Getting a product to "production" quality is less than 50% of the effort required for commercial vendors. Making profit from closed-source software is not easy.

11) MIT has already become famous amongst the digirati as the birthplace of GNU and FSF. It is one of the "hallowed halls." Enjoy it while you're there.

Lastly, I want to add that while there is a strong notion, particularly amongst the Linux community, of open and fair advocacy, not all users abide by this, and flame mail may be generated. As with all communities, a spectrum of opinion and temperament exists. Don't take the flames from the bad apples to heart, and don't judge the community as a whole on the actions of a few hotheads.

From Prevelakis Vassilis <vp@unipi.gr>:

I was appalled to read Matthew Craighead's letter.

I do not think that slashdot.org represents all Open Source proponents. In fact until I read about that particular site in the letter, I was oblivious to its existence.

Now, more to the point, I am afraid Matthew made a logic jump. slashdot IS pro-OSS AND slashdot IS crazy DOES NOT IMPLY pro-OSS IS crazy.

For example, I am sure that there exist Web sites run by fascists or other extremists. If I visit such a site I may also find material about how great America is. Does this imply that anybody who is pro-American agrees with the rest of the material in that site?

I am pro-OSS so my opinion may not count, so let's look at what a staunch proponent of capitalism such as the *Economist* magazine has to say about OSS ("Computer programming. Hackers rule," 20-Feb-99; see <www.economist.com>).

Open-source programming is more like academic work than business. And just as the disclosure of theories and empirical data usually produces good science, so published code leads to better software. The programmers are motivated not chiefly by money, but by reputation. It is a coup to write "patches" that pass the scrutiny of fellow hackers and get incorporated in the next version of a program. . . . Yet there are drawbacks. Big software companies have every reason not to go

more letters...

open-source. Hackers might also not be keen to work alongside the likes of IBM and Sun; many are strongly anti-commercial. There is also the danger of forking when a group falls apart and incompatible versions of a program emerge as has happened to one operating system, BSD Unix, when personality conflicts led to splits. Few managers will bet their companies on the product support they receive in news groups on the Internet.

Hence the importance of the commercial fringe to open-source software. Numerous service companies, such as Caldera, Red Hat and S.u.S.E, have built a business out of making Linux easier to install. Eric Allman, the benevolent dictator of Sendmail, has set up a company that supports the open-source development of the program, while selling a commercial version and services to support it. . . . It is too early to say whether such approaches will work. But open-source is here to stay. Perhaps the software industry will eventually look a bit like a highway. The infrastructure (operating systems, networking technologies) will be largely a public good, while services (support, training) and specialised applications are for sale. Just don't expect Bill Gates to like the idea.

I find myself in full agreement with that article. Besides, capitalist dogma also states that the market should be left to decide what is best. So let's wait until the "horribly large number of Linux fanatics" at MIT graduate and join companies and then we'll see whether their adoration for OSS is commercially justified.

As far as the tone of the letter is concerned, I think that if people disagree with any concept, they can express arguments against it. Calling the other side communists or other names does not help the discussion.

From Yiorgos Adamopoulos

<adamo@dblab.ece.ntua.gr>:

It is clear to me that Matt has not worked to earn his living as a CS/CEng professional. What Matt seems to be missing is that people support OSS because it is better than corporate software (in many cases, not all, not even most; I have yet to see something that competes with Excel, for example).

What also Matt seems to miss is that in any ideology (and OSS is an ideal) there exist fanatics (like the ones he saw in the slashdot posts). Working for "love and fame" requires a total change of the day-to-day model that the world has. In such a world clothing/eating/housing is a solved problem. I for one do not believe that we are going to have such a world ever (the ancient Greeks didn't make it and it was easier then ;-).

Matt, your motives in life show in your letter: You are into money and fame. Well, simply put, not all humanoids are. You seem to miss the fact that if the other side "wins" the "race" there will be a total change in the society, so the logic of "starving," "no TV," and "no radio" do not apply ("no TV" is a benefit to the society anyway).

Ask yourself how Linus Torvalds makes a living today – if he needs more – and enjoy your university years, because they will undoubtedly be the best in your life. Also learn to give to others, you will always get more back (that is what OSS is really about IMNSHO).

From Marty Leisner <leisner@rochester.rr.com>:

Matthew Craighead's letter bothers me. While lots of people says outrageous things about OSS (or Free Software as it used to be called), lots of people say outrageous things about everything – from politicians to televangelists to the guy on the street corner – you have to separate the wheat from the chaff in unmoderated

discourse and quote people who have useful opinions.

While many consider Richard Stallman's views to be somewhat socialist, I don't recall him espousing anti-capitalist views. While he called for a software tax (which I don't support) his attitude is very simple: "I consider that the golden rule requires that if I like a program I must share it with other people who like it." He later says, "By working on and using GNU rather than proprietary programs, we can be hospitable to everyone and obey the law" (GNU Manifesto, 1985, <<http://www.fsf.org/gnu/manifesto.html>>).

It turns out a small percentage of the world's software industry write "off the shelf" applications. Most software work is inhouse labor. What I find maddening is that proprietary applications rarely provide the source code. From working with software for two decades, I know that having the source code (it helps if it's logical and good quality) can often make sense out of a knotty problem. If the problem isn't commonplace, support is impossible remotely, the only advice to give (after "Is the computer plugged in?") is "Maybe reinstall the application." I wonder if the cost to everyone of managing proprietary applications exceeds the cost of writing the proprietary applications. I found it humorous (or pathetic) when several days were lost at work due to a mysterious Word virus.

With binary applications, the only way to debug "problems" is via tools like strace or strings (I've seen countless instances where set-uid-non-root applications cannot read files which exist – due to permissions). They generally give useless diagnostics (of the order "Cannot read important file").

Having good quality source code is often a much more productive way to use a computer. If you regularly do something which the software doesn't properly support, having source code allows the source code to be changed. And if you

more letters...

can't change it, you can hire someone to do it. Commercial off-the-shelf software vendors rarely do work for hire ... it's not in their model.

Richard Stallman has a model of selling support and service for a profit. The actual software should be free. I'm not sure I go for that, but if I buy an application, why can't I have the software for a nominal fee (let's say several times the price of the application). Whenever I ask a vendor, "How much is the source code," they're mystified. Oftentimes they quote a price several hundred times the price of the product. Other times they say, "It isn't for sale" (it's an attitude the baffles me ... I can understand them insisting on NDA, but "not for sale?").

Also, if Linus Torvalds developed Linux with the idea of selling it, it wouldn't have become a usable system. I understand Linus is doing very well financially, and he has substantially more than 15 minutes of fame.

I'm also baffled by developers (for example, the xforms library), who make binaries freely available without providing source code. I find it hard to justify spending time with software (which invariably has bugs) without source code — so if I want to I can fix a problem or at least understand it.

OSS and Linux

From Steven Lembark <lembark@wrkhors.com>:

In the last issue of *login*: [April 1999], Rik Farrow wondered (mused?) whether commercial distributions of Linux might prove its bane. The example cited is HP's arcane shadow password system. The (rather accurate) description was: "Yikes!" Now for some good news: (a) passwords have nothing to do with the "Linux" kernel itself and (b) the problem can be fixed because if it's Linux then you have the source. Point (a) matters: so

long as HP doesn't botch the kernel too badly everything on top of it can be fixed.

Here is an alternative outcome:

HP distributes their system, using their shadow scheme. The password setup drives people crazy. In order to make the scheme work HP hacked the `fgetpwentq(3)` and friends of `pwd.h` to handle their lookups. With all these people not liking the hacked `libc` distributed with HP's Linux there is a market for a "clean" version. So, someone starts with `libc` from GNU, makes it work with HP's Linux, compiles the utilities with it, and distributes a shadow-in-a-box-like package. Let's say it only costs \$500 and they sell only 500 copies worldwide: \$250,000 isn't chickenfeed.

Open-source software may be the way to keep hardware vendors honest. They can't botch the software too badly without someone else *fixing* it. They can't even screw up Linux too badly because the source is available to fix. If they can all agree on Linux then we might actually have hardware companies in the hardware business and software companies in the software business!

The one worm in this apple would be another "distribution" war, à la window-manager battles. So long as Linux stays open and keeps evolving, however, we have a chance to get a reasonably standard OS. And fixes.

Software Patents

From Simon J. Gerraty <sjg@quick.com.au>:

I saw Cynthia Deno's article in the recent *login*: [April 1999] and just thought I'd point out that much of the software industry (myself included) thinks that the best improvement that could be made by the US Patent Office would be to abolish software patents altogether.

The issues mentioned in *login*: regarding the difficulties surrounding software patents are good arguments in favor of the above position.

Of course that would upset a lot of lawyers ...

more...

Correction

Figure 4 on page 56 of the February 1999 issue of *login*: (Vol. 24, No. 1) was incorrectly printed, cutting off both top and righthand edges. Our apologies to Jeffrey Mogul; here is a correct version of figure 4:

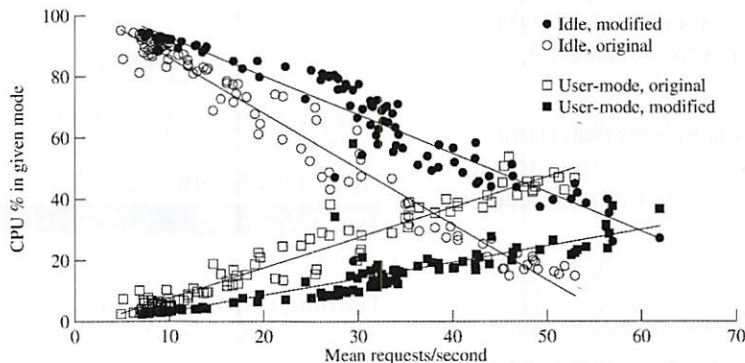


Figure 4. CPU costs as a function of request rate

The USENIX Crossword Puzzle

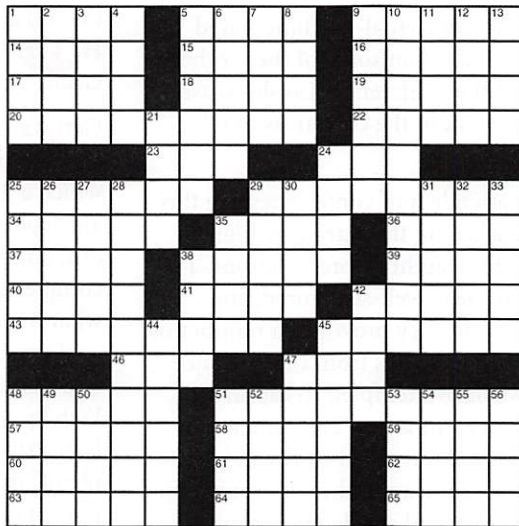
Across

1. Fight
5. Juan's house
9. Seize power
14. Intoxicating aussie root
15. Ray Steven's Ahab was one
16. Asian country
17. A holly
18. Found on a finger or toe
19. Struck sharply
20. Bad transportation idea
22. Ringworm, e.g.
23. One or more
24. Mop up
25. High performance CPU architect
29. Reason for life's existence
34. At full speed
35. Kolstad's curse
36. String+ball weapon
37. Pokes fun at
38. Nun outfit
39. Bring program to memory
40. Yarn verb
41. Sendmail author
42. Inverted commas
43. Resonant
45. Make wealthy
46. Sheep brain+tape-worm disease
47. Matter (legal)

48. Oyster product
51. Digital signature methodology
57. Of hearing
58. Free of disease
59. Pennsylvania port
60. Illegal deed
61. Unhappy utterance
62. Light hue
63. Cooked bread
64. Playthings
65. Bible book

Down

1. Slip
2. Next of: palf, palf, palf
3. Declare positively
4. Devito TV show
5. Doglike
6. Sheik place?
7. Stanford's answer to Lisp
8. Can
9. Free from obstruction
10. Somewhat open
11. Atop
12. Fixed ratio
13. Defendant's answer
21. Volume control label
24. Parasitic fungi; internet plague
25. German bills
26. Type of acid
27. Polio dude
28. Frequency distribution graphs



29. Ancient Egyptian measure
30. Of the ear
31. Cavity holder
32. Fill with joy
33. Basement gas menace
35. Mountain lake
38. Brain carrier
42. Bit carrier
44. Small lunar valley
45. Grade schooler's coffee-break
47. Servomotor
48. International treaty
49. New money in Holland
50. Opera solo
51. Internet talk
52. Surrounds Ken's head
53. Bristly plant organ
54. Liquid waste acid type
55. Coin factory
56. Tennis game units

P	O	M	P	S	T	R	U	T	S	T	A	Y
E	P	E	E	C	A	U	S	E	A	R	G	O
R	E	S	T	R	A	I	N	E	R	F	A	U
I	N	S	T	A	L	L	S	M	O	A	N	E
W	A	T	F	O	R	A	M	E	R	I	C	A
O	R	I	O	N	C	R	E	S	S	R	U	M
R	I	N	G	C	U	R	L	S	C	I	T	E
M	E	T	M	O	L	A	L	C	O	B	O	L
S	L	O	V	E	N	L	Y	F	O	R	E	S
M	I	N	T	S	P	A	N	T				
T	S	E	T	S	E	M	A	R	G	I	N	A
I	O	T	A	S	P	E	R	M	A	C	E	T
D	U	E	L	T	E	R	S	E	A	M	O	S
E	R	R	S	S	P	E	E	D	L	A	P	P

Solution to the April Puzzle



conference reports

This issue's reports focus on the 3rd Symposium on Operating Systems Design and Implementation (OSDI '99), held in New Orleans, LA, on February 22-25, 1999.

See <http://usenix.org/publications/library/proceedings/osdi99/> for the full program of technical papers from this conference.

Our thanks to the summarizers:

David Sullivan

Keith Smith

Xiaolan Zhang

Zheng Wang

Rasit Eskicioglu



Margo Seltzer & Tynan

3rd Symposium on Operating Systems Design and Implementation (OSDI '99)

NEW ORLEANS, LOUISIANA

February 22-25, 1999

Overview by David Sullivan

The third OSDI was, in the words of program chairs Margo Seltzer of Harvard University and Paul Leach of Microsoft Research, designed to "define the charter of operating-systems research for the coming decade" and to address whether OS researchers should be "embracing more divergent areas." The keynote about the World Wide Web by Jim Gettys and a lively panel on virtual-machine-based systems touched on some of these other areas, but the conference also showcased excellent work in the core areas of OS research.

Veteran attendees of conferences like this one remarked on the extremely high quality of the authors' presentations. The talks were clear, well-structured, and engaging, and they provoked a number of thoughtful questions from the audience which we have attempted to capture in the session summaries. The conference featured a well-attended works-in-progress session, a number of evening BOF sessions, and ample opportunities for attendees to socialize and exchange ideas while enjoying the conference-sponsored receptions, as well as the cuisine and local color of New Orleans's renowned French Quarter.

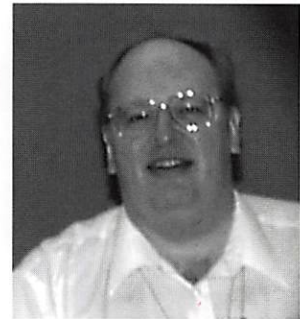
If the papers presented at the conference can be considered a foretaste of what is to come, there is an abundance of important work to be done during the coming decade of operating-systems research. And OSDI, which in its third instantiation was declared an established tradition by the program chairs, will be there to continue to showcase that work.

KEYNOTE ADDRESS

The Blind Men and the Elephant

Jim Gettys, Compaq Computer Corp. and the World Wide Web Consortium

Summary by Keith Smith



Jim Gettys

Jim Gettys is a senior consultant engineer for Compaq Computer Corporation's Industry Standards and

Consortia Group and a visiting scientist at the World Wide Web Consortium (W3C) at M.I.T. He is the chair of the HTTP/NG Protocol Design Working Group of W3C.

Gettys's talk took its title from the John Godfrey Saxe poem of the same name, in which a group of blind men encounter an elephant and each man, touching a different part of the elephant, draws a completely different conclusion about what manner of beast they've met. By analogy, Gettys suggested that any attempt to understand or to optimize the Web by considering only one component is probably doomed to failure. The "elephant" of the Web consists of many components with strong interactions between them. To further complicate matters, all of these components are changing.

The two most significant parts of the Web are what happens on the wire (i.e., HTTP), and the content – HTML, style sheets, images, Java applets, etc. There are numerous interactions between these parts, for example, between Web content and the protocols that are used to access it, or between content and caching. Legal and social interactions are also interesting.

Gettys described HTTP as a "grungy" protocol: verbose making poor use of TCP, and failing to separate metadata and

content. Version 1.1 of HTTP, however, now being deployed, addresses many of these problems. It allows for persistent connections and the pipelining of requests. It supports content encoding, for compression. When HTTP 1.1 is well implemented, one TCP connection carrying HTTP 1.1 outperforms four concurrent connections carrying HTTP 1.0. HTTP 1.1 also has better support for caching.

Naturally, the changes in HTTP 1.1 lead to some interesting interactions. As an example, Gettys discussed the interaction between TCP and data compression, observing that compression scales up faster than the savings in bytes on a high-speed network. With cache validation performing 2-5 times better in HTTP 1.1, Gettys also speculated that this would change applications.

Web content is also changing. The advent of style sheets offers a variety of benefits, reducing the need for repetitive markup and ad hoc images. Style sheets will also reduce overhead by decreasing both the raw number of bytes that need to be transmitted and the number of HTTP requests, since many design elements that are now embedded images can be described more tersely with style sheets.

Other changes in Web content include the move from GIF to PNG images, new content types such as vector formats, and content types that are seldom used at present because of bandwidth constraints (e.g., audio and video).

The next topic Gettys addressed was the caching and proxy infrastructure. He observed that much of the so-called "dynamic content" in the Web could be cached, as the databases from which it is generated are only updated periodically. Gettys cited data that shows, contrary to some predictions, that the fraction of dynamic content in Web traffic is not increasing.

Currently most caching is done at the periphery of the Net, near the clients.

Gettys argued that caching would be more effective if it also occurred in the middle of the Net; the closer a cache is to the server, the more of that server's load the cache can offload. Web caching also has interesting interactions with intellectual-property issues. Can servers trust a proxy not to give their data to the wrong people? This has obvious importance for pay-per-view content.

Another area of interest is the increasing use of HTTP as a transport. More and more metaservices are being implemented on top of HTTP. Frequently, this involves using forms to invoke functionality on remote servers. Gettys pointed out that posting a form is equivalent to a method invocation on a remote object, or like an `ioctl()` call, without a procedure signature. This is a hole you can drive an elephant through. Current object-oriented technology is too brittle. In the Internet, either end must be able to change independently. In particular, there is no way for such a metaprogram to know when the underlying form has changed. As a result, the metaprogram might inexplicably stop working, or it could start doing undesirable things such as ordering thousands of dollars of products you don't want.

People frequently think that HTTP can be used to tunnel through a firewall. While this might work sometimes, firewall administrators weren't born yesterday. The firewall can look at content before passing on a request. If they don't know what is in it (e.g., SSL), they won't let it through.

HTTP is getting extended in all sorts of ways. CORBA, DCOM, Java RMI, and a variety of other protocols are now being run on top of HTTP. The result is frequently poor performance. DCOM and CORBA were originally designed for use on a local network and are even more verbose than HTTP.

Changes in the technology used for local loops will also have an impact on the future of the Web. With the advent of

DSL and cable modems, traditional modem technology is dead. A variety of other technologies may also come into play in providing the final link to the user – satellites (e.g., Direct TV), data over 110/220 volts (power companies already have a right of way to your house), and noncellular wireless. Gettys pointed out that the explosion of wireless devices means that bandwidth will still be a concern for the foreseeable future, as these devices often have less bandwidth than today's dial-up modems.

The many changes in the components of the Web, and the complex interactions among them, lead to some questions. Will application developers optimize for speed, or will they be content to keep download time constant? As new facilities become available, will site designers use them? Will future improvements lead to faster sites or to more junk on the page? Gettys feels that tools are the key to the future. Current tools are terrible, frequently generating excessively verbose and invalid HTML. Many current tools don't support important new technologies, such as caching.

In closing, Gettys observed that we are all neophytes. Researchers working with the Web, himself included, are starting to get a sense of the shape of the elephant, but still need to understand the interactions of the various parts before optimizing any single part in isolation.

In the Q&A session, Paul Leach of Microsoft suggested that Gettys's HTTP/NG work suggests that he must have some opinions about the answers to the questions that he closed his talk with. Gettys replied that fundamentally, it's about metaservices. They are being used by more and more programs and make safe extensibility vital. As the Internet evolves, things need to break at the right times.

Greg Minshall of Siara Systems asked where pressure can be applied to make tools better, and whether there is an economic pressure on the tool suppliers to

provide better ones. Gettys replied that there should be economic benefits to making tools easier to use and to making the end-user experience better. He also observed that it is difficult now for tools to support cascading style sheets (CSS), as neither Netscape nor Internet Explorer supports them fully, and they implement different subsets of CSS.

Session: I/O

Summary by Keith Smith

Automatic I/O Hint Generation Through Speculative Execution *[Best Student Paper]*

Fay Chang and Garth A. Gibson,



Fay Chang

Carnegie Mellon University

Fay Chang presented this work, one of two winners of the award for Best Student Paper. It was one of those wonderfully novel papers that presents a seemingly bizarre idea that turned out to work surprisingly well.

The research was performed in the context of the Transparent Informed Prefetching (TIP) system that Hugo Patterson presented at the 1995 SOSP. In that system, applications were manually modified to provide file-prefetching hints to the kernel. Chang and Gibson's work eliminates the need for manual modification by providing the prefetching hints automatically through speculative execution of the application. The basic idea is that when an application blocks on a read request, a second thread in the same application (the "speculative thread") continues executing, only instead of issuing read requests, it issues prefetching hints.

One of the major concerns about adding this speculative thread to an existing

application is ensuring program correctness. Chang addressed this concern by noting that their system does not allow the speculative thread to execute any system calls except the hint calls, as well as `fstat()` and `sbrk()`. Exceptions generated by the speculative thread are ignored. To prevent the speculative thread from modifying data values that the main application thread needs, Chang and Gibson use "software-enforced copy-on-write." Before each store, the code determines whether the target memory region has been copied yet. If not, a copy is made. In either case, the store (and subsequent loads) are redirected to the copy. To insulate the main thread from the performance impact of these extra checks, they make a complete copy of the application's text and constrain the speculative thread to execute in that copy.

They maintain a log of the hints generated by the speculative thread. When a real I/O request is generated by the application, they check that the request matches the next hint in the log. If it doesn't match, they halt the speculative thread, tell the operating system to ignore any outstanding hints, and restart the speculative thread from the current location of the application. This technique allows the speculative thread to catch up if it falls behind and also allows it to get as far ahead of the main thread as possible as long as it is generating accurate hints.

Chang next discussed SpecHint, a tool that she and Gibson developed to generate the speculating binary by rewriting the original binary, thus avoiding the need to access application source.

Finally, she presented the results of experiments conducted to evaluate the system. They used three of the test programs from the TIP benchmark suite — `xDataSlice`, `agrep`, and `gnuld`. For all of these programs, the speculating version showed improved performance when compared to the original nonhinting version. In addition, two of the speculating versions showed performance improvements comparable to those achieved by

programs with manually inserted hints from the original TIP work.

To measure the overhead of speculating, they ran the speculating versions of their test programs on a system with prefetching disabled. They saw a 1-4% slowdown compared to the unmodified versions of the test programs.

In the Q&A session, Fred Douglass of AT&T Research asked Chang to elaborate on what happens when the speculating thread wants to execute a disallowed system call. Chang replied that for most calls, the system call stub is replaced by code that returns success.

Jochen Liedtke of IBM's T.J. Watson Research Center asked why Chang had chosen to use a software-based copy-on-write scheme rather than a traditional hardware-based approach. Chang replied that they had tried forking a speculative version of the program, but they found the restart costs (i.e., `fork()`) prohibitive.

IO-Lite: A Unified I/O Buffering and Caching System *[Best Paper]*

Vivek S. Pai, Peter Druschel, and Willy Zwaenepoel, Rice University



Vivek Pai

This paper won the conference's Best Paper award.

Vivek Pai started by observing that network-server throughput affects many people's perceptions of computing speed, because for them it's crucial to end-user response time.

A problem with current operating systems is that they contain many independent buffers and caches in different layers of the system: the filesystem buffer cache,

VM pages, network mbufs, application-level buffers, etc. The interactions between these buffers and caches cause two problems – data copying and multiple buffering – both of which degrade overall system performance.

The goal of IO-Lite is to unify all caching and buffering in the system, allowing applications, the network, the filesystem, and IPC to use a single copy of the data safely and concurrently. Concurrent access to the data is accomplished using immutable shared buffers. Programmers manipulate the buffers using a data structure called a buffer aggregate, which provides a level of indirection to the physical data. This technique is similar to Fbufs, which were presented at the 1993 SOSP by Peter Druschel.

IO-Lite was implemented in a general-purpose operating system, FreeBSD, as a loadable kernel module. The IO-Lite API includes two new system calls, `iol_read` and `iol_write`, which are similar to the generic `read` and `write` system calls, except that they operate on buffer aggregates. The API also includes routines for manipulating buffer aggregates.

Pai pointed out that since IO-Lite's buffers are immutable, the combination of the physical address of a buffer and its generation number gives you a unique identifier for the data in the buffer. This UID can be used to cache information about the buffer. The network system, for example, uses this technique to cache the checksum for a buffer.

In comparison testings Flash-Lite outperformed the authors's Flash Web server, typically by 40-80%.

Kevin Van Maren from the University of Utah asked whether the IO-Lite buffers are pageable, and if so, what happens when the network tries to DMA to/from one. Pai replied that the IO-Lite buffers are pageable, but for network access they pin the pages in memory. Jose Brustoloni from Lucent's Bell Labs observed that some applications assume a specific lay-

out of data in memory, and asked whether IO-Lite would support such applications without performing a data copy. Pai replied that in such cases they would need to perform one copy.

An attendee from Sandia National Labs asked how IO-Lite handles cache replacement. Pai said that IO-Lite buffers are reference counted. If there are no references to a buffer (other than from the cache), it can be replaced. The VM system pages things out normally, using LRU. An attendee from Veritas Software asked how IO-Lite would work on a system in which file data and file metadata coexist in the same cache. Pai replied that in their implementation platform, FreeBSD, a separate static file cache is used only for metadata. He didn't see any problem, however, using IO-Lite on systems that use the same cache for both, although you would probably want to pin metadata pages down separately.

Virtual Log Based File Systems for a Programmable Disk

Randolph Y. Wang, University of California, Berkeley; Thomas E. Anderson, University of Washington, Seattle; David A. Patterson, University of California, Berkeley

Randolph Wang opened his talk with a simple question, "How long does it take to write a small amount of data to a disk?" An optimist would consider the time to transfer the data from the head to the disk and answer, "20 microseconds." A pessimist would take into account the costs of seek and rotational latencies and answer, "Several milliseconds." The goal of this work was to deliver microsecond write performance to applications and make it scale with disk bandwidth.

Wang explained that the problem with traditional filesystems is that the interface between the host filesystem and the disk controller is limited in expressive power, and the I/O bus doesn't scale up. Their solution is to move part of the filesystem into the disk, taking advantage of the

CPU power available on today's disks and exploiting the free bandwidth there.

The authors minimize the latency of small synchronous writes by writing them to free sectors or blocks near the current location of the disk head. They call this technique "eager writing." To make it work, the disk maintains a table mapping logical blocks to their physical locations. This table is also written using eager writing. They handle recovery by threading the different pieces of the table together into a "virtual log." The log is a backward chain, with each record containing a pointer to the previous log record. In the event of power failure, all the system needs to do is to write the tail of the log to disk. Wang said that engineers at disk vendors had indicated that it would be easy to modify the disk firmware to perform this write to a fixed location prior to parking the heads.

Since disk support for eager writing does not yet exist, the authors used a disk simulator to evaluate their system. Pei presented a comparison of a standard implementation of the UNIX File System (UFS) to UFS running on a virtual-logging disk (VLD), as well as of LFS and LFS running on a VLD. The results show a substantial improvement in the performance of small-file creation and deletion when the VLD was used.

Since the performance of eager writing depends on the availability of free disk space near the disk head, the authors evaluated the performance of the different test systems for a variety of disk utilizations. Both UFS systems showed a slight performance degradation as the disk filled. Although LFS performed excellently at lower utilizations, its performance degraded much more quickly as the disk filled.

Sean O'Malley of Network Appliance observed that virtual logging moves a piece of the filesystem onto the disk. As a result, you have two filesystems, one on the disk and one on the host machine. O'Malley asked whether the two filesystems

tems wind up fighting with each other. Wang replied that he had really only moved a piece of information to the disk – whether or not the log is free. If you want to move more functionality onto the disk, you probably need to redefine the disk interface.

Peter Chen from the University of Michigan wanted to know how reliable writing the tail of the log to disk during powerdown is. Wang replied that the people he spoke with thought it was reasonable to write as much as a track of data at powerdown.

Session: Resource Management

Summaries by Xiaolan Zhang

Resource Containers: A New Facility for Resource Management in Server Systems *[Best Student Paper]*

Gaurav Banga and Peter Druschel, Rice University; Jeffrey C. Mogul, Compaq Western Research Laboratory



Gaurav Banga

This paper was the other winner of the Best Student Paper award. Gaurav Banga began by discussing the motivation behind the work: the fact that general-purpose operating systems provide inadequate resource-management support for server applications. In current systems, processes serve as both resource principals and protection domains. The joining of these two roles is often inappropriate for server applications. In a Web server, for example, many independent tasks share the same process, and much of the processing associated with HTTP connections happens inside the kernel and is unaccounted for and uncontrolled. Banga

and his co-authors developed the resource container abstraction to separate the notions of protection domain and resource principal, and thus enable fine-grained resource management.

Resource containers encompass all of the resources used by an application for a particular independent activity. The system associates scheduling information with a resource container, not a process, thus allowing resources to be provided directly to an activity, regardless of how it is mapped to threads. To be effective, resource containers require a kernel-execution model in which kernel processing can be performed in the context of the appropriate container.

The authors implemented resource containers in Digital UNIX 4.0. They modified the CPU scheduler to implement a hierarchical decay-usage scheduler that treats resource containers as its resource principals. They also modified the network subsystem to associate received network packets with the correct resource container, allowing the kernel to charge the processing of each packet to its container.

Banga presented results showing that resource containers are quite lightweight, and he discussed two experiments designed to test their effectiveness. In the first, one high-priority client and several low-priority clients request documents from a Web server. Without resource containers, the response time for the high-priority client increases greatly as the number of low-priority clients increases because of added networking processing in the kernel. With resource containers, the response time also increases, but in a much more controlled way.

In the second experiment, a Web server's throughput for static documents was measured in the face of an increasing number of CGI requests. Without resource containers, the throughput of the static requests decreases dramatically as the number of CGI requests increases. But resource containers can be used to

create a "resource sandbox" around the CGI connections, allowing the static throughput to remain constant.

Banga concluded by emphasizing that resource containers are purely a mechanism and are general-purpose in nature. A lot of recent scheduling work can be used in conjunction with resource containers.

In questions following the talk, Eric Eide of the University of Utah sought to confirm that resource containers do not provide an authorization scheme. Banga said that this is indeed the case; resource containers are orthogonal to protection. Eide then asked if, when issuing a read from a file, you need to build a new container or can just use the default container with which you're associated. Banga said that either approach could be used. Timothy Roscoe of Sprint, addressing a point also raised by Mike Jones of Microsoft Research, pointed out that schemes like this one traditionally encounter problems when a server (such as an X server) and its clients are on the same machine and thus share the same resources. He asked how resource containers would be used in such cases. Banga replied that resource containers just provide a mechanism, and application-specific policies need to be built on top of them. Michael Scott of the University of Rochester said that he was puzzled by the criteria used to decide what goes into a resource container; things seem to be grouped together that are not logically coherent. He wondered if resource containers could be used for logically coherent things and then given different amount of resources using something like lottery scheduling. Banga agreed that this could be done.

Defending Against Denial of Service Attacks in Scout

Oliver Spatscheck, University of Arizona; Larry L. Peterson, Princeton University

Oliver Spatscheck started by asking why denial-of-service (DoS) attacks are a con-

cern. The short answer is the Internet, and the situation is getting worse since routers now allow third-party code to run. Spatscheck outlined a three-step process to defend against DoS attacks: (1) account for all resource usage; (2) detect violations of your security policy; and (3) revoke resources when violations occur.

Spatscheck presented the Scout system, which allows this three-step process to be implemented in the context of special devices that attach to the Internet, such as routers, firewalls, WWW servers, and other network appliances. Since Scout was designed with such devices in mind, it addresses the need to support soft real-time, but does not provide complete, general-purpose OS support.

Scout builds a network appliance by combining modules, such as a disk module, a filesystem module, HTTP, TCP and IP modules, and a network device-driver module. Scout also introduces the concept of a path, to which distributed resources are bound. Paths contain global state pertaining to a particular I/O data flow, and each of them has its own thread pool. Paths provide automatic mapping of shared memory, as well as automatic thread migration along the modules contained in the path.

Spatscheck also described Escort, the security architecture for Scout. Escort allows the modules that have been configured into a system to be isolated into separate protection domains. A configurable resource monitor is responsible for resource protection and accounting. All resource usage is charged to an owner, which can be either a path or a protection domain. A DoS attack is detected by a violation of the configurable security policy. The resource monitor can deal with it in one of three ways, depending on how the policy has been configured: (1) suspend data delivery; (2) deny resources; (3) destroy the owner.

The authors implemented a Web server in Scout with two test configurations – one protection domain per module, and

all modules in the same protection domain. Accounting overhead is about 8% for Scout with a single protection domain, and a factor of four for six protection domains. All interrupts and almost all cycles were correctly accounted for. With a SYN flood attack of 1000 syns/second, regular clients slowed down by only 5-15%.

At the conclusion of the presentation, one audience member asked if Spatscheck had any ideas or mechanisms for defending against domain-directed attacks, as opposed to path-directed ones. Spatscheck said that one method would be to have multiple instantiations of modules. For example, you could have two different IP networks; if one is corrupted, you can replace that protection domain with another one.

David Black from EMC asked how their system can distinguish between good and bad packets in situations in which IP sources are forged. Spatscheck provided three ways of addressing this: use firewalls to block forged IPs; authenticate IP addresses; let it go and see if it violates the policies in place. The last approach might have a larger performance impact than the other two, but you can still revoke all of the resources after detection.

Greg Minshall from Siara Systems asked how their I/O buffers compared with IO-Lite. Spatscheck explained that one writer creates an I/O buffer and, once a reader locks it, the writer loses its privileges. The advantage over IO-Lite is that Scout's path abstraction tells you the protection domains into which a buffer should be mapped.

Self-Paging in the Nemesis Operating System

Steven M. Hand, University of Cambridge

Steven Hand began by explaining that the goal of his work is to support simultaneously both continuous media/soft real-time applications and more traditional,

“standard” applications. He noted that conventional operating systems offer poor support for several reasons. First, they schedule the CPU using priority schemes which tell you *who* should get the processor, but not *when* or *how much*. Second, contention for other resources is arbitrarily arbitrated. Third, there can be “QoS crosstalk” as a result of the kernel's performing a significant amount of work on behalf of applications. In particular, applications that repeatedly cause memory faults will degrade overall system performance.



Steven M. Hand

Hand's work, requires every application to deal with all of its own memory faults using its own concrete resources. “Self-paging” involves three principles: (1) control – resource access is multiplexed and resources are guaranteed over medium-term time scales; (2) power – high-level abstractions are not imposed on the underlying resources, giving applications greater flexibility; and (3) responsibility – applications must carry out their own virtual-memory operations.

More specifically, self-paging requires that the system grant/allocate physical frames explicitly, dispatch all memory faults to the faulting application, allow applications to map/unmap their own pages, and provide low-latency protected access to the backing store.

The fourth requirement is fulfilled by the user-safe backing store (USBS). The USBS is composed of the swap filesystem, which is responsible for admitting an application into schedule and allocating it some region of the disk for swap space,

and the user-safe disk, which schedules I/O requests according to their associated QoS.

Hand mentioned that one lesson he learned in doing this work is that exposing low-level details works better on RISC architectures. The results on an x86 are very poor, in part because of the higher kernel/user boundary crossing overhead. In addition, he mentioned that while granting each application its own disk time slice allows applications to optimize their own access patterns, large seeks may be required every time you switch between applications. It's possible to do better globally if you allow applications to interleave. It's an open research question as to what the tradeoffs are between global performance and local optimizations, and how to get the best of both worlds.

Miche Baker-Harvey from Equator Technologies asked about how truly shared resources (e.g., a page fault on a shared library) are handled. Hand said that applications involved in sharing might need to deal with a third party. For shared libraries, a system service charges everyone a fair share. I/O can also involve a lot of sharing; when DMA is used, you need to pin down the memory. Jay Lepreau of the University of Utah asked if you could have several protection domains cooperating on a task and sharing the paging responsibilities among them. Hand said that they could and pointed out that Nemesis's activation domains are orthogonal to protection domains, and they can be assigned to multiple accountable entities.

Eric Cota-Robles of Intel asked about the level of granularity used for reservations in the system, as well as how reservations for different resources interact. Hand pointed out that in choosing the level of granularity, there is a tradeoff between quality and overhead. He chose 250ms because that works as well as 100ms and 500ms; for anything smaller than 100ms, the overhead of context switches becomes

unacceptable. As far as different resources are concerned, he mentioned that Nemesis uses EDF for both disk and CPU. Allocating resources independently and making sure they interact well is tricky; it is something they are still working on.

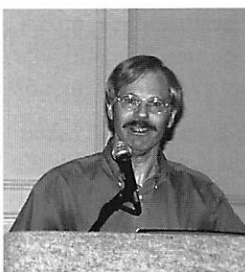
Finally, Bruce Lindsay from IBM pointed out that we've talked about external pagers for almost 10 years (about 45 Web years). He asked if, with all this talk, there had been any commercial utilization of external pagers. Hand gave as an example any system based on Mach.

Sources and documents for this system are available at
<<http://www.cl.cam.ac.uk/Research/SRG/netos/nemesis>>.

Panel: Virtual Machine-based Operating Systems

Summary by Zheng Wang

Panelists: Ken Arnold, Sun Microsystems Inc.; Thorsten Von Eicken, Cornell University; Wilson Hsieh, University of Utah; Rob Pike, Bell Labs; Patrick Tullmann, University of Utah



Paul Leach

Moderator Paul Leach introduced the topic of the panel discussion: what's new and what's not in virtual-machine-based operating systems. Today, when someone talks about a virtual machine (VM), it is usually based on the Java language. In days of yore, however, the language could be Smalltalk, Lisp, or Pascal, among others. Is Java just another programming language, or has it introduced something new to OS research?

Ken Arnold defined a virtual machine as a system with a uniform instruction set, uniform system calls, uniform libraries, uniform semantics, and a uniform platform. He then presented his view of a

computer as a system hooked to the network instead of a system based on local disks. With network connections, the Internet can become "my computer." Compared to "your computer," "my computer" is bigger; it grows geometrically and gets better exponentially. Meanwhile, "my computer" breaks more often (when the remote resource fails) and gets fixed more easily (when a similar resource is found elsewhere on the network). Also, "my computer" can be "our computer." For this situation, VM-based OSes are the only solution. To take advantage of "my computer," each node should provide the code for using its service, that is, the ways to talk to this node. This is not particular to Java, but a Java VM is an example of a homogeneous system over a network. Finally, Arnold claimed that "everything else is wasted research." He qualified the statement by saying that "wasted" in this case does not mean "useless." His point was that there are only a small group of people working on VMs compared to the number of questions to be answered, and that is a wasted opportunity.

Thorsten Von Eicken started by comparing the traditional, virtual-memory-based OSes with new, virtual-machine-based OSes. He noted that the concepts of page-level memory protection, user/kernel privilege, and hardware primitives in virtual-memory-based OSes are comparable with the concepts of object-level protection, module/class privilege, and type-system primitives in virtual-machine-based OSes. "What's new" in virtual-machine-based OSes includes the Java language, real protection (against malicious code), resource management (e.g., enforcing limit, revocation, termination), and safe-language research. Von Eicken said what's interesting here is the balance between sharing and resource management. This introduces a lot of trade-off possibilities among program compile time, link time, load time, and run time. Pitfalls ("showstoppers") include Java's speed, garbage collection, debugging, and the need to design

around the Java class loader. In summary, Von Eicken stated that virtual-machine-based OSES complement (but don't replace) virtual-memory-based OSES and provide a new playground for revisiting old OS issues.



L. to R.: Ken Arnold, Rob Pike, Wilson Hsieh, Patrick Tullman, Thorsten Von Eicken

Wilson Hsieh started by drawing a comparison between virtual machines and operating systems. Since both VMs and OSES are layers above the hardware on which the applications run, Hsieh concluded that an OS is already a VM. So what are the distinctions between VMs and OSES? According to Hsieh, one issue is multiple-language support. OSES usually support many languages, while VMs typically don't. Another is whether to do certain tasks, such as protection and resource management, in software or hardware. OSES generally provide a way for users to talk about resources. This is exemplified by the functionality of C language. VMs and their associated languages typically hide resource management from users. VMs can either let the underlying OS do the work or handle it themselves. In the latter case, could the VM be smarter than most of today's OSES?

During Hsieh's talk, questions were raised about the performance of VMs, especially that of the existing Java VM, compared to native OSES. Ken Arnold argued that the performance of the Java VM is sufficient, considering that most people have fast computers. He pointed out that Java bytecode can now run up to twice as fast as native code. Some audience members suggested that Java's speed problems

come from memory-hierarchy performance and application load time.

Rob Pike of Bell Labs had a fairly different point of view. Pike was a leader on the Inferno project, in which a virtual machine was integrated with the operating system underneath. Pike listed several advantages of the VM approach: It provides real portability; programs can

be compact (the VM itself is large, however); it allows a single address space with the VM providing protection; integrating the VM and OS eliminates the overhead of the VM entering the kernel; and the OS becomes a runtime library for the VM and provides resource management. However, Pike's conclusion after the project was that we should not merge OSES and VMs. He cited a number of reasons: there is no possibility of compartmentalization since the memory, resources, and execution of the VM and OS are intermingled in horrific ways; debugging is a nightmare; and the storage models of the two are incompatible, as are their process models (scheduling schemes will be either separated and hard or mingled and messy).

The last panelist, Patrick Tullmann, observed that a VM runtime is comparable to that of an OS. So the question is, why do we bother with VMs, and if we structure VMs as OSES, where is the win over hardware-based OSES? Tullmann gave two answers. One is fine-grained sharing, where computers share not only data but also code. The second is optimal resource accounting. One example is a malloc-less server/kernel, where the server resources are allocated by the client and passed down to the server.

During the Q&A session, David Black of EMC Corporation commented that the panelists' talks sounded like a "solution seeking a problem." He asked the panelists to name the real problem. Tullmann's answer was fine-grained sharing (and the fact that graduate students need something to work on). Arnold answered with "plug and play." He used a real-life example of having to look for the Windows CD-ROM in order to install a new printer, suggesting that the printer should be able to handle itself. Pike said the problem is supposed to be portability. Von Eicken pinpointed security problems from running untrusted code, but Pike disagreed, questioning the necessity of downloading methods through a VM. Hsieh said the problem lies in structuring software.

Session: Kernels

Summary by Rasit Eskicioglu

Tornado: Maximizing Locality and Concurrency in a Shared Memory Multiprocessor Operating System

Ben Gamsa, University of Toronto; Orran Krieger, IBM T.J. Watson Research Center; Jonathan Appavoo and Michael Stumm, University of Toronto

Orran Krieger began by pointing out that the performance of a simple multithreaded counter operation (increment/decrement) on today's modern shared-memory multiprocessors is orders of magnitude worse than on older systems, and argued that OSES for modern systems should be designed in a fundamentally different way. The main goal of Tornado is to maximize locality and thus reduce this performance problem.

Tornado uses an object-oriented design; every physical and virtual resource in the system is represented by an object. These objects encapsulate all the data structures and locks necessary to manage the resources. This approach has two key benefits. First, encapsulation eliminates any sharing in the underlying operating

system. Second, an object-oriented approach allows multiple implementations of objects, enabling the system to choose the best implementation for a given situation at runtime. Additional innovations in Tornado that help maximize locality and concurrency include clustered objects, protected procedure calls, and semi-automatic garbage collection.

Krieger next described the experimental platform. The reported results are based on a 16-processor NUMA prototype currently being developed at the University of Toronto, as well as the SimOS simulator from Stanford. Further experiments were performed on some modern commercial multiprocessor systems. Tornado demonstrated large performance gains on the microbenchmarks. Future research will address scalability and running real applications. The core technology has already been incorporated into the Kitchawan OS from IBM Research, which runs on PowerPCs and x86 processors.

Finally, Krieger summarized some of the lessons they had learned. If used carefully, an object-oriented strategy is good, and the cost of this approach is more than compensated for by the locality achieved. Also, indirection with translation tables is a useful tool. Furthermore, an object-oriented strategy, in conjunction with clustered objects, allows fine-tuning of the system using incremental optimization.

Marvin Theimer from Microsoft Research asked Krieger to give a sense of the percentage of the Tornado kernel that they found impossible to parallelize in the manner described in the talk. Krieger replied that they hadn't encountered anything that couldn't be parallelized. He pointed out that there was an important tradeoff here. In Tornado, they get a big win for policies that need only local information to make decisions, but there may be areas where they will lose because they can't make global policy decisions.

Interface and Execution Models in the Fluke Kernel

Bryan Ford, Mike Hibler, Jay Lepreau, Ronald McGrath, and Patrick Tullmann, University of Utah

Jay Lepreau discussed the implementation and API of Fluke, a microkernel-based operating system motivated by nested virtual machines. A process is able to implement some OS services for its children and have the rest taken care of by whoever provides those services to the parent.

Lepreau described two models of execution, the "process model" and the "interrupt model." In the process model of execution, each thread of control has a kernel stack, and a blocking thread's state is implicitly stored on its stack. Most monolithic kernels, such as Linux, UNIX, and Windows NT, fall into this category. On the other hand, in the interrupt model of execution, there is only one kernel stack per processor, and the required thread state is explicitly stored in the thread control block (TCB) when the thread blocks. This category includes systems such as V, QNX, and the Exokernel implementations.

All conventional kernel APIs belong to the process model. They support long-running operations and maximize work per kernel call. In this model, thread states are inexact or unobtainable. In the interrupt model (a.k.a. "atomic" APIs), thread states are always well defined and visible, and per-thread kernel state is minimized.

The Fluke kernel exports an atomic API while also supporting long-running operations. It can support both the process and interrupt execution models through a build-time configuration option. The basic properties of the Fluke API include promptness, correctness, and completeness, as well as interruptible and restartable kernel calls. These properties greatly facilitate services such as user-level checkpointing or process migration.

They also simplify permission revocation and facilitate application development. Unfortunately, the atomic API has some disadvantages: extra effort is needed to design it, intermediate system calls are needed, and there is extra overhead from restarting system calls.

Concerning performance, Lepreau addressed preemption latency, rollback overhead, and speed. As expected, a fully preemptive kernel (only possible in the process model) always allows much smaller and predictable latencies. Non-preemptive kernels for both models exhibit highly variable latencies causing a large number of missed events. On the other hand, even with only a single preemption point, the preemptible interrupt model fares well on the benchmarks he discussed. Similarly, the rollback cost during a page fault is very reasonable compared to the already high cost of page-fault handling. As an unoptimized, experimental kernel, Fluke does not show any major slowdowns for any of the five execution model/preemptibility combinations.

Lepreau concluded that an atomic API is easy to implement and that OS folks can do as well as "those hardware guys" who provide fully interruptible "long" instructions such as block move.

After the talk, Margo Seltzer of Harvard said that she was reminded of the Lauer and Needham paper that discussed the equivalence of message-passing and shared-memory OS architectures, so she was expecting to see similar conclusions about the interrupt and process execution models. Lepreau replied that if the API and implementation are done right, there is an equivalence between these two models. However, the two models can have performance differences, as he and his colleagues discovered.

Jon Shapiro of IBM observed that both EROS and KeyKOS have an atomic API and that 25% of the restart cost for those systems was the user-to-supervisor reentry. He was curious to know where Fluke

restarted (i.e., in user mode or kernel mode). Lepreau said that Fluke restarts in user mode, just on the other side of the system-call boundary.

The Fluke sources are available at <http://www.cs.utah.edu/flux/>.

Fine-Grained Dynamic Instrumentation of Commodity Operating System Kernels

Ariel Tamches and Barton P. Miller, University of Wisconsin

Ariel Tamches described his research on runtime kernel instrumentation. His vision for future operating systems is that they should provide a dynamic, unified infrastructure that allows fine-grained runtime code instrumentation for purposes of performance measurement, tracing, testing and debugging, optimization, and extensibility. Such an infrastructure would provide measurement primitives, such as simple counters, cycle timers, and cache-miss counters, that could be used to instrument the kernel as it runs. Similarly, it would allow functions on certain code paths to be analyzed using predicates. Using these measurements and runtime code-insertion technology, the kernel would be able to be optimized dynamically.

KernInst is a fine-grained, dynamic kernel-instrumentation tool that allows users to insert runtime-generated code into an unmodified Solaris kernel. Using code splicing, the machine-code instruction at an instrumentation point is overwritten with a jump to a code patch that consists of the runtime-generated code, the overwritten instruction, and finally a jump back to the following instruction.

Code splicing has some inherent problems. For example, jumping to a patch using two instructions cannot be done safely. A context switch at the wrong time could lead to the execution of the original first instruction followed by the new second instruction. However, the address range that is reachable within a single instruction on a SPARC is limited to ± 8

megabytes. This problem is solved by introducing an intermediate branch location, called a springboard. When the patch code is far away, the instrumented instruction is replaced with a branch to a springboard that contains a long jump with as many instructions as needed to reach the code patch. In general, any scratch space located close to the splice point in the kernel is suitable for a springboard.

Tamches next described a simple kernel-measurement tool that he built on top of kerninst as a proof of concept. This simple tool counts the number of calls to any kernel function as well as the number of kernel threads executing within a kernel function. This tool was used to analyze the performance of the Squid v1.1.22 proxy server. Since it seemed that the L2 cache functionality of the disk was not performing well, the performance of the I/O routines in the kernel were analyzed. Analysis revealed that the open function, which was called 20-25 times/sec took 40% of time and was the real bottleneck. Within open, the name-lookup and file-create functions were the two sub-bottlenecks. Squid creates one file per cached HTTP object in a fixed hierarchy of cache files. It also reuses stale files to eliminate file-deletion overhead. However, before overwriting the files, Squid was truncating them first, which caused UFS to synchronously change the metadata. Two simple modifications to Squid eliminated these bottlenecks – the size of the directory-name lookup cache in the Solaris kernel was increased, and the Squid code was modified to truncate the file only when needed.

Steve Pate from Veritas Software asked what the runtime overhead of the instrumentation extensions was. Tamches replied that it was the (additional) overhead of adding two extra branches and one cache miss to your code. Bruce Lindsay from IBM Research asked how an optimized version of a routine would work once installed. Tamches replied that

an optimized routine could be downloaded as just another code patch. The initial routine would then be instrumented at its entry point to check for the proper conditions, and to jump to the optimized version if the conditions are satisfied.

Marvin Theimer from Microsoft Research asked how complicated it was to write the patch code. Tamches observed that it could be difficult for complicated patches. So far they haven't done anything more complicated than the counters and timers presented in this work, although they do have some experience with doing other types of patches using Paradigm, a tool that performs the same types of operations on user-level applications.

Marianne Lent from Veritas software asked whether it was possible to unload instrumentation points after they were spliced into the kernel. Tamches replied that this could be done by restoring the instructions that were overwritten in installing the instrumentation points.

Session: Real-Time

Summary by David Sullivan

ETI Resource Distributor: Guaranteed Resource Allocation and Scheduling in Multimedia Systems

Miche Baker-Harvey, Equator Technologies, Inc.

Miche Baker-Harvey described the ETI Resource Distributor (ETI RD), a scheduler designed for use on multimedia processors, which allows you to emulate fixed-function hardware such as MPEG video encoders and decoders, audio devices, and modems. Since the system must maintain the illusion that real hardware is present, this scheduler must support what Baker-Harvey termed "firm" deadlines that are harder than conventional soft realtime guarantees.

Baker-Harvey characterized the types of applications that the ETI RD was designed to support. They: (1) are pri-

marily periodic; (2) can shed load if the system becomes overloaded; and (3) have mainly discrete resource requirements that are known long in advance (e.g., processing an MPEG frame to a given resolution requires a known amount of CPU time). She argued that existing approaches for soft realtime scheduling are inadequate for this type of environment. Reservation-based schedulers provide firm guarantees but do not allow for graceful load shedding; constraints and best-effort schedulers handle overload but do not provide strong guarantees.

She next outlined the three components of the ETI RD. First, a Resource Manager performs admission control and grant control, deciding whether a task can be given scheduling guarantees and what percentage of each resource will be given to each task. An application requests admission by giving the Resource Manager a resource list consisting of the resource requirements for each level of quality of service (QoS) that it can provide. Each entry includes a function that can be called to provide that level of QoS. The Resource Manager admits a task if the sum of its minimal requirements and the minimal requirements of all currently admitted tasks can be simultaneously accommodated.

The Resource Manager also computes a grant set that gives applications the largest possible resource share that the system can accommodate. This grant set is passed to the second component of the system, the Scheduler, which uses an EDF algorithm. In overload, the Resource Manager calls the third component of the system, the Policy Box, which passes back a policy that determines which applications should shed load and by how much. Together, these three components guarantee that an admitted task will be scheduled every period until it exits, and that its allocations will be from among the ones defined by the task.

Baker-Harvey went on to explain some of the finer details of the system. Finally, she

addressed performance. Context switches take a reasonable amount of time and are only taken when necessary. Admissions and grant control are done in the context of the task that needs the computations to occur, so that other tasks' guarantees are not impacted. She also described an experiment in which a set of threads with various possible levels of QoS gradually request admission to the system. As each additional thread is added, the system becomes more and more overloaded, and the grant set is adjusted to give each thread a smaller grant.

A Feedback-driven Proportion Allocator for Real-Rate Scheduling

David C. Steere, Ashvin Goel, Joshua Gruenberg, Dylan McNamee, Calton Pu, and Jonathan Walpole, Oregon Graduate Institute

Ashvin Goel presented a scheduler designed for "real-rate" applications like software modems, Web servers, and speech recognition tasks, whose throughput requirements are driven by real-world demands. Current priority-based schedulers are inflexible and ill suited to fine-grained allocation, whereas reservation-based schedulers require the correct specification of the proportions needed by each application and fail to provide adequate dynamic responsiveness. The system that Goel discussed addresses these problems by using a feedback-based scheme to dynamically estimate the proportion and period needed by a given job, based on observations of its progress.

Goel described how their system is able to estimate application progress through what he termed "symbiotic interfaces," which link application semantics to system metrics. For example, a queue shared by a producer and a consumer could use a symbiotic interface that exposes the queue's size and fill level and the role of each thread. The kernel can then monitor the queue's fill level and adjust the allocations given to the producer and the consumer as needed.

Goel then explained the role of the feedback controller, which first computes a pressure for each real-rate thread based on its progress metrics. The pressure is fed to a proportional-integral-derivative (PID) control which determines the allocation given to the thread. Realtime threads with known reservations can specify allocation and period directly, and miscellaneous jobs are treated as if they had a constant positive pressure. When the allocations determined by the controller lead to overload, it "squishes" the allocations of real-rate and miscellaneous jobs using a weighted fair share approach where the weighting factor is an importance associated with each thread. Realtime jobs with specified reservations and real-rate jobs that are driven externally are given a "quality exception" so that their resource reservations can be renegotiated.

In the performance section of his talk, Goel discussed two experiments that tested the controller's responsiveness by having a producer (with a fixed reservation) oscillate between two rates of production, one double the other. The controller succeeded in adjusting the consumer's allocation so that its rate of progress closely matched that of the producer. Goel also mentioned tests that show that the controller overhead is linear in the number of threads, but with a small slope. He concluded with a brief discussion of related work and future directions.

Jose Brustoloni of Lucent/Bell Labs asked about the sensitivity of the system to the choice of parameters. Goel replied that a dispatch interval of 1 ms and a controller period of 10 ms seem to work well, and that more work needs to be done on setting the PID parameters and on determining if one unique set of parameters works for all applications.

Gopalakrishnan of AT&T Labs raised the possibility of application-specified progress metrics being used in denial-of-service attacks by tasks that fraudulently claim to have made no progress at all.

Goel agreed, but pointed out that quality exceptions provide a mechanism for applications to figure out that such attacks are occurring, and policies can manipulate job importance to insulate a thread from an attack.

Carl Waldspurger of Compaq Systems Research Center asked how the user-specified notion of importance interacts with the progress metric in determining allocations. Goel responded that their current system provides a set of dials that users can adjust to dynamically control the importance of various tasks, and that other approaches (such as economic models) could also be used.

A Comparison of Windows Driver Model Latency Performance on Windows NT and Windows 98

Erik Cota-Robles and James P. Held,
Intel Architecture Labs

Erik Cota-Robles began with the motivation for this work: Realtime and multimedia applications are becoming increasingly common on personal computers, and general-purpose operating systems are thus being asked to provide temporal correctness, which depends on the time-complexity of the calculation, hardware and OS latency, and concurrent resource utilization. Throughput metrics are insufficient for such applications. He defined the notion of an application's latency-tolerance, which depends on the amount of buffering that it does (since before a deadline is missed, all buffered data must be consumed) and is orthogonal to how processor-intensive the application is. He also pointed out that on Windows, in addition to interrupt and thread latency, there is an additional latency from deferred procedure calls (DPCs), which are used by interrupt service routines (ISRs) for compute-intensive computations.

Cota-Robles went on to describe the design goals used in developing their microbenchmarks to measure latency.

They wanted to achieve near-zero measurement overhead to avoid embedding the benchmarks in loops, and they wanted to cover a variety of behaviors with a few tests. The Pentium timestamp register allowed them to achieve a single-instruction measurement cost. They used the programmable interval timer as the source of hardware interrupts, and they measured the latency to the software ISR, to the DPC, and to the kernel-mode thread. On NT, the first of these measurements cannot be made, since the timer ISR cannot be modified. While their methodology was developed for Windows, Cota-Robles mentioned that it could also be applied to UNIX.

Next, Cota-Robles covered the measurement methodology used in this work. Since latency measurements are uninteresting on a quiescent system, they used a spectrum of application stress loads. For repeatability, they used the Winstone97 benchmarks, a number of 3-D games, and a set of Web-browsing benchmarks that included viewing files of various sizes as well as audio/video playback using RealPlayer.

The resulting distributions had very large tails. As a result, Cota-Robles and Held focused on the median, and they also characterized the thickness of the tail in terms of hourly, daily, and weekly worst-case values.

Cota-Robles pointed out that for Windows NT there is almost no distinction between DPC latencies and thread latencies for threads at high realtime priority. On Windows 98, on the other hand, there is an order of magnitude reduction in the worst-case latencies that a driver obtains by using DPCs as opposed to realtime high-priority kernel-mode threads.

Cota-Robles presented some additional data on Windows 98 thread latency. Finally, he briefly looked at an example involving a soft modem to illustrate how the latency numbers gathered by their

tools can be used to reason about quality of service even before an application is available.

Victor Yodaiken of New Mexico Tech commented that it is not always the case that OS latency swamps the hardware effects; on realtime Linux, the effect of going to the timer over the ISA bridge is the dominant effect. He also asked why external timing wasn't used in this work. Cota-Robles said that the latency tolerance of the applications they were interested in was much greater than the timer tick rate; they didn't care about anything under a millisecond.

Session: Distributed Systems

Summary by Xiaolan Zhang

Practical Byzantine-Fault Tolerance

Miguel Castro and Barbara Liskov,
M.I.T.

Byzantine-fault-tolerant systems are hacker-tolerant – they can continue to provide correct service even when some of their components are controlled by an attacker. Hacker-tolerance is important, because industry and the government increasingly rely on online information systems, and current systems are extremely vulnerable to malicious attacks.

Research on Byzantine-fault tolerance is not new, but most of it has demonstrated only theoretical feasibility and cannot be used in practice. The few techniques that have been developed for practical application make unrealistic assumptions such as synchrony and are too slow to be useful.

Miguel Castro presented a Byzantine-fault-tolerant replication algorithm that doesn't rely on synchrony assumptions, performs faster than previous implementations, and is resistant to denial-of-service attacks. He also discussed a replication library based on this algorithm that he and Barbara Liskov used to implement BFS, a Byzantine-fault-tolerant NFS ser-

vice. Using the Andrew benchmark, they showed that BFS is only 3% slower than the standard NFS implementation on Digital UNIX.

Castro pointed out that the algorithm can be used to implement any deterministic replicated service. It provides two main properties. First, it ensures that the replicated system behaves like a correct, centralized implementation that executes operations atomically one at a time – a strong safety property called linearizability. Second, it provides liveness, which ensures that the service remains available despite faults. The algorithm relies on several assumptions, including the existence of at least $3f+1$ replicas to tolerate f Byzantine faults, and the presence of strong cryptography.

The algorithm is a form of state-machine replication. The authors use a primary-backup mechanism to maintain a total order of the requests to the service. Replicas move through a succession of configurations called views. In each view, one replica is the primary and the others are backups. The primary assigns a sequence number to every request, and the backups check on the primary and ensure that it is behaving correctly. When the primary misbehaves, the backups trigger a view change to select a different primary.

Castro also discussed some optimizations that they implemented. To reduce the cost of large replies, only one client-designated replica sends the full result; the others just send a digest of the result. In addition, replicas can tentatively execute a request as soon as the request prepares, which improves latency. For read-only requests, clients multicast requests to all replicas and wait for $2f + 1$ replies with the same result, retransmitting as necessary. Finally, the performance of message authentication was improved using message-authentication codes for all messages except view-change and new-view messages, which still use slower digital signatures.

The authors are working on a number of extensions, including how to recover faulty replicas and support for fault-tolerant privacy.

Marvin Theimer from Microsoft Research said that they were comparing something that writes synchronously to disk with something that doesn't. He asked what the overhead would be for a replicated service that didn't need the disk. Castro said that the paper also presents a comparison with an unreplicated system that did not write to disk, and that the overhead was 26%. Theimer then asked if turning the power off on all replicas risked corrupting their filesystem. Castro concurred, and Barbara Liskov suggested that using a UPS could prevent this. Peter Chen from the University of Michigan asked if Castro could give an example of the class of faults their system is targeting. Castro said any class of attacks, as well as non-deterministic software errors. Burton Rosenberg from Citrix asked if, when the public key is replaced by the MAC secret key, the secret keys have to remain secret even though the faulty nodes know them all and can broadcast them. Castro replied that there is a secret session key between every active client/replica pair.

The Coign Automatic Distributed Partitioning System

Galen C. Hunt, Microsoft Research;
Michael L. Scott, University of Rochester

Galen classified distributed systems into what he called the 5PM of distributed systems: people, protection, peripherals, persistence, processes, and memory. If an application needs any two of these and they're located on different computers, then it needs distributed software. A fundamental problem of distributed computing is to decompose the distributed software into pieces and to decide where to place them. Smart programmers can do it statically. But static partitioning is expensive; the optimal partition can be

user- or data-dependent, and it changes with the network topology.

Coign is an automatic distribution-partitioning system for applications built out of COM components. The authors use scenario-based profiling to quantify the communication between the components and the application, and analyze that information to partition and distribute the application with the goal of minimizing the distributed communication. All this is achieved without access to source code.

Hunt explained that the process takes four steps: (1) take the binary of the application and find the objects inside of it; (2) use scenario-based profiling to identify the interfaces between those objects; (3) quantify the communication across the interfaces; and (4) build a graph and cut the graph to produce an optimal distribution.

Hunt then showed a live demo of Coign using PhotoDraw (an image composition application that will ship with Office 2000). First, he instrumented the binary to intercept every COM call. Next, he ran the application on a training dataset. There was a 45% overhead for the instrumented version of the software. In the worst case, it could be as high as 85%. Next he took the profiling information, combined it with the network statistics, and created a graph for the program. The graph was a giant circle, with each dot on the circumference representing a COM object. Lines connecting the dots were COM interfaces. The idea was to put objects that communicate heavily on the same machine. The graph algorithm computed a distribution model. Finally, a distributed version of PhotoDraw was created. Galen pointed out that when the distributed version runs on two machines, it's 25% faster than the original version. Since the source code of PhotoDraw is 1.8 million lines, it would be hard to analyze manually.

Galen also looked at another application, MSDN Corporate Benefit. He was able to

reduce the communication time by 35%. Ironically, this application was originally designed as a model of good distributed programming techniques.

One key issue in Coign is identifying similar transient objects across multiple executions. When the application creates an object, it needs to figure out which object it corresponds to from the profiled scenarios, so that it can decide where it should be located. This has to be done dynamically. The key insight is that similar objects across executions have similar instantiation histories. The algorithm walks the stack and creates an instantiation signature consisting of callers and objects. The goal of the classifier is to identify as many unique objects as possible, since the more unique objects it can identify, the more distribution choices it has.

Galen pointed out that the idea of automatic distributed partitioning is not new. The contribution of Coign is its application of automatic partitioning techniques to dynamic, user-driven applications. Several open questions remain, including how to handle distributed error handling, how to merge vertical load balancing across an application with horizontal load balancing, and how to achieve live repartitioning.

Chris Small from Bell Labs asked if they had thought about implementing different metrics for deciding how to do partitioning, since, for example, minimizing network communication isn't the same as minimizing overall latency, because machines run at different speeds. Hunt replied that it would be simple to add processing speed to the Min-Cut/Max-Flow graph-cut algorithm that he used, but that including memory consumption (and anything that cannot be converted directly to time) is an open research question.

Session: Virtual Memory

Summary by Rasit Eskicioglu

Tapeworm: High-Level Abstractions of Shared Accesses

Peter Keleher, University of Maryland

Distributed shared-memory (DSM) protocols support the abstraction of shared memory to parallel applications running on networks of workstations. The DSM abstraction facilitates programming, allowing users to avoid worrying about data movement, and it allows applications to become portable across a broad range of platforms. Unfortunately, the DSM abstraction does not allow applications to improve performance by directing data movement. The proposed extensions to overcome this problem are usually protocol-specific and not portable across multiple platforms.

To address this problem, Peter Keleher developed the "tape" mechanism. Tapes are objects that encapsulate accesses to shared data. They allow applications to hide or eliminate data-access latency and to get all the benefits of data aggregation by moving data in advance of demand. The tapes can be used in various ways once they are created ("recorded"). For example, the tapes can be added to messages and applied ("played back") when they are received. Tapeworm is a library of such tape operations.

Keleher explained that a tape consists of a set of events, each of which is described by an ID number identifying an interval of a process's execution and the set of page IDs that were accessed during that interval. Several operations can be applied to a tape, including adding and subtracting pages and "flattening" it into an extent of pages mentioned by the tape.

Keleher next described how the tape mechanism interacts with the underlying DSM system. Hooks into the underlying consistency protocol allow tapes to be generated by capturing shared accesses, while hooks into the message subsystem

are used for such things as adding tapes to messages or extracting them at the other end. These hooks are used to construct the Tapeworm library, which provides three types of synchronization mechanisms: (1) update locks, (2) record-replay barriers, and (3) producer-consumer regions.

Keleher indicated that the emphasis of Tapeworm was on being able to create high-level descriptions, rather than on actual performance gains. Nevertheless, Keleher reported performance improvements enabled by the tape abstraction. For the six applications in the test suite that he used, speedup improvements averaged close to 30%. On average, the number of messages sent was reduced by 50%. Furthermore, the average of remote-miss improvements was 80%. Keleher concluded that the tape mechanism is cheap, effective, and easy to use, especially as part of a run-time library. Future work includes encapsulating consistency semantics into objects.

Margo Seltzer of Harvard University mentioned that this work reminded her of the Eraser system presented at the last SOSR, and she asked if the tape abstraction could be used as a debugging aid for multithreaded programs. Keleher agreed with the suggestion and indicated that one of his students used the underlying data-movement mechanism to detect data races in distributed applications.

MultiView and Millipage: Fine-Grain Sharing in Page-Based DSMs

Ayal Itzkovitz and Assaf Schuster, Technion-Israel Institute of Technology

Ayal Itzkovitz briefly talked about the first software DSM system, IVY, and identified two major performance limitations: page size (false sharing) and protocol overhead (high number of messages). Over the years, there have been many attempts to overcome the false-sharing problem. One common approach is to relax the consistency requirements. This

approach lessens the problem but does not totally eliminate it. It changes the semantics of memory behavior, introduces memory-access and processing overhead, and is still a coarse-grain solution. A newer approach is to provide compiler support and do code instrumentation. This technique solves false sharing, but it isn't portable, it has substantial run-time overhead, and, in some cases, it requires additional hardware support.

Close examination of these problems identified two improvement possibilities: (1) reducing the sharing unit (i.e., page size), and (2) using ultra-fast networking and thus minimizing protocol overheads. Itzkovitz described a new technique called MultiView for implementing small-sized pages (mini pages). Multiview is highly efficient and does not require a special compiler. In addition, it offers application-tailored granularity and resolves false sharing with practically zero overhead. It is built as a "thin protocol layer" on top of the operating system. The basic idea is to map each shared variable to a different, nonoverlapping virtual-memory region, called a view. In reality, these variables may reside on the same (physical) page, but they are not necessarily shared. Furthermore, each variable may have different access privileges. MultiView lets the application manage each of these views independently as needed, with separate access policies.

Itzkovitz next described Millipage, a software DSM system based on the MultiView technique. Millipage is implemented as a user-level shared library on top of Windows NT 4.0. The programming model of Millipage is sequential consistency, which also allows application-tailored fine access control. It employs a static-home approach for each minipage. Fast Messages from the University of Illinois is the underlying communication mechanism on a Myrinet LAN. The `malloc()` system call is wrapped to allocate a new minipage each time it is called.

Basic costs in Millipage are quite low. For example, invoking a remote page takes less than 300 microseconds, including all software processing. A suite of five applications was used to test the performance of the Millipage DSM system on a testbed cluster of 8 PCs with Pentium II processors. All applications showed encouraging speedups, although there was a tradeoff between false sharing and aggregation for some applications.

Itzkovitz concluded that MultiView approach performs as well as any other relaxed model. In future work, using the compiler to do the minipage allocation and mapping, and improving access locality with the help of the operating system, will be investigated, as will revisiting relaxed consistency models and considering the applicability of MultiView approach to other services, such as global memory systems, garbage collection, and data-race detection.

After the talk, Michael Scott from the University of Rochester admitted that he was wrong when he claimed about ten years ago that false sharing was the problem for software DSM. He added that nobody wants to use software DSM for applications with 8MB datasets, but, rather, for applications with gigabyte datasets in which fine-grain sharing is impractical. If there is no significant false sharing, then aggregation becomes a big issue for performance. He suggested that the minipage approach is very promising for addressing pathological cases where only a small portion of the data requires fine-grain access.

Optimizing the Idle Task and Other MMU Tricks

Cort Dougan, Paul Mackerras, and Victor Yodaiken, New Mexico Institute of Technology

Dougan indicated that this project grew out of an effort to optimize the PowerPC port of the Linux operating system. The major constraint was to get good performance without breaking Linux compati-

bility. This implied that the efforts should be concentrated on the architecture-specific components of Linux. Memory management was the obvious starting point.

Dougan and his colleagues discovered that the OS was occupying one-third of the TLB entries on average. The PowerPC offers an alternative translation, called block address translation (BAT), from logical to physical that bypasses the TLB mechanism. The idea of using superpages to reduce TLB contention was not practical for a straightforward implementation using BAT, because there were only 8 BAT registers. Since user processes are generally ephemeral, only the kernel memory was mapped using the BAT mechanism. This approach reduced the percentage of TLB slots used by the kernel to nearly zero. Also, a 10% decrease in TLB misses and a 20% decrease in hashtable misses were observed during the benchmarks. The real measure of this improvement was a 20% decrease on a complete kernel compilation time.

Since the PowerPC uses a hashed page table (HTAB), the next optimization attempt was to improve the efficiency of this HTAB. A three-level table is used to back the HTAB, and it is searched on a HTAB miss. The next idea was to reduce hashtable hot spots by scattering the entries in the hashtable. A different virtual segment identifier (VSID) generation technique was used to reduce collisions.

Dougan next indicated that their original conjecture that TLB reload speed was not as important as reducing TLB misses was incorrect. This suggested yet another improvement opportunity. The HTAB miss-handling code was rewritten in assembly and executed with the MMU turned off. Also, the assembly code was optimized to reduce pipeline stalls. As part of this optimization, the HTAB is completely eliminated on the PowerPC 603 by performing searches only on Linux tables. These efforts yielded major performance improvements: 33% reduc-

tion in context switching and 15% reduction in communication latency as measured with *lmbench*. This optimization strongly reduced the effect of BAT mapping on both the 603 and 604.

The best optimization was achieved by tuning the idle task. Since the OS must provide cleared pages to users, this functionality is moved to idle task to be performed during its execution. Large performance gains were obtained when the cache was off during this operation. The full kernel compilation was done in 5.5 minutes. *lmbench* also showed 15% across-the-board latency improvements. Dougan concluded that memory management is extremely sensitive to small changes. Small operations in the kernel can destroy locality. Also, intuition is not reliable, microbenchmarks are needed, and repeatable sets of microbenchmarks like *lmbench* are invaluable. He added that OS designers need to learn from processor designers about quantitative measures. As the second part of his conclusion, Dougan said that some version of superpages can help, and that page-fault-handling details are critical for good performance. More work on cache preloading is needed, and greater use of dynamic code generation will be pursued.

One attendee from Bell Labs asked if one can generalize by saying that it is better to have a software-based TLB and cache management than to have hardware-managed versions. Dougan replied yes, because a software approach usually gives greater flexibility.

Session: Filesystems

Summary by Zheng Wang

Logical vs. Physical File System Backup

Norman L. Hutchinson, University of British Columbia; Stephen Manley, Mike Federwisch, Guy Harris, Dave Hitz, Steven Kleinman, and Sean O'Malley, Network Appliance, Inc.

Norman Hutchinson began by giving a brief motivation for this work. Since

filesystems are getting bigger and disks are getting bigger still, ensuring the reliability of data stored on filesystems is becoming more and more difficult.

Hutchinson outlined two basic backup approaches they studied and compared: logical backup and physical backup. Logical backup is a file-oriented approach, such as that employed by the UNIX *dump* and *tar* commands. The advantage of this approach is portability, because the formats used are platform-independent. In physical backup, on the other hand, the data of one physical medium is replicated on another physical medium. Although the physical strategy is nonportable, it is very fast, because the medium is accessed sequentially. Also, the output is an exact copy of the original. UNIX's *dd* command and Plan 9's filesystem backup strategy are examples of this approach.

Before going into the details of their implementations of these two approaches, Hutchinson described Network Appliance's Write Anywhere File Layout (WAFL) filesystem. WAFL is a log-based filesystem that uses NVRAM to reduce latencies. WAFL stores metadata in files, the most important of which is the block-map file, a generalization of the free bitmap that indicates what blocks are being used for what purposes. WAFL also uses copy-on-write techniques to provide snapshots, read-only copies of the entire filesystem. A disk block referenced by a snapshot is never overwritten.

Hutchinson described their logical backup implementation as an evolution of the BSD *dump* utility. After finding the files to be backed up, the program does a sequential i-node traversal because the files are written to backup media in increasing i-node order. This restrictive requirement is augmented in their implementation by a custom pre-fetch policy at the kernel level. Their logical restore first creates an in-core virtual directory tree and then uses this tree to resolve any name lookup that needs to be done, thus

avoiding the creation of unnecessary directories and increasing the speed of lookups. Their physical backup implementation first writes a snapshot of the filesystem and then simply walks through the blockmap file sequentially and writes all blocks that are in use (i.e., referenced by the snapshot). For speed purposes, the physical-backup process bypasses the filesystem. With physical restore, some cleanup is necessary.

The performance of these implementations was measured using an aged filesystem, basically a physical copy of the company's live filesystem, to reflect real-life situations. The first set of measurements was collected by doing a single-tape (both logical and physical) backup and restore of a 188GB filesystem. Logical backup took 7.4 hours, with a speed of roughly 7.8MB/sec, while logical restore requires 8 hours with a speed of 8.8MB/sec. Physical backup and restore has only a single stage of simply reading and writing the disk blocks, which takes 6.2 hours for backup and 5.9 hours for restore. Interestingly, logical backup and restore require significant amounts of CPU, whereas the physical operations have marginal CPU requirements.

In order to improve performance for extremely large filesystems, it is necessary to add more tapes to increase backup bandwidth. This is tricky for logical backup, because the format of a dump tape is fixed and cannot easily be spread across several tapes. Therefore, multiple dumps are performed in parallel. On the other hand, spreading the dumps across multiple tapes is easy for physical dump, since all the disk blocks are independent. Using a four-tape implementation, logical dump requires about 3 hours and logical restore requires 4 hours, while physical dump and restore operations each completed in a little under 2 hours.

Hutchinson concluded by pointing out that the physical strategy provides better performance, whereas the logical strategy is more flexible. Therefore, two remain-

ing areas of research are making logical backup faster and making physical backup more flexible.

Rob Pike of Bell Labs asked if Hutchinson was familiar with the Plan 9 papers on filesystem-backup strategies, and he further asked why one couldn't restore a single file on a standalone file server. Hutchinson replied that he was familiar with the Plan 9 papers, and that the operation is tricky because the metadata kept on the backups is totally independent of the disks, and thus additional information is needed to interpret the metadata.

Masoud Sadrolashrafi of Veritas Software asked if snapshots are sufficient for restoring data in different situations. Hutchinson replied that snapshots are sufficiently self-contained for it to be possible to rebuild a filesystem from any snapshot.

The Design of a Multicast-based Distributed File System

Bjorn Gronwall, Assar Westerlund, and Stephen Pink, Swedish Institute of Computer Science and Luleå University of Technology

Bjorn Gronwall introduced JetFile, a distributed filesystem targeted at personal-computing environments and designed for ubiquitous file access over local- and wide-area networks with different physical characteristics. The system relies on a model in which clients are also the servers in the system. Taking a protocol-centric approach to distributed-filesystem design, JetFile's major goals are to hide the effects of delays induced by propagation, retransmission, and limited bandwidth, and to minimize and localize the traffic. In order to achieve these goals, JetFile uses optimistic algorithms to increase update availability of files and to hide update delays. It also uses replication and multicast to increase read availability of files. Using clients as servers also decreases update latencies and improves scalability. Other methods to

reduce latencies, such as hoarding and prefetching, are planned for future work.

JetFile uses Scalable Reliable Multicast (SRM) layered on top of IP multicast. SRM's communication paradigm consists of two kinds of messages: request and repair. SRM is a receiver-oriented protocol, in which the receiver makes a multicast request when it needs data and those who are able to respond to the request (i.e., have the data requested) send a repair message containing the data. In order to eliminate the inflation of repair messages from many hosts, each host that is able to respond sets an internal timer and waits. If a repair message for the same data is received, the node cancels the timer. Otherwise, the host sends the repair message when the timer expires. The value of the timer is randomized with a bias toward the closer hosts.

Files in JetFile are named using tuples, such as organization, volume, file number, and file-version number. All tuples except the version number are hashed to map files to multicast channels. Therefore, a particular file always uses the same multicast channel. The basic JetFile protocol deals with data units, which can be either status objects (carrying file attributes) or data objects (carrying actual file contents). Correspondingly, SRM messages include status-request, status-repair, data-request, data-repair, plus version-request and version-repair for retrieving file-version numbers.

To retrieve file contents in JetFile, the receiver node multicasts an initial data-request message, and the source node responds with a multicast data-repair message. Since the receiver now knows the source of the data, the remaining data objects are transferred using the same protocol but with unicast request and repair messages. File updates use write-on-close semantics. Since the client acts as a server for the new file version, a lot of write-through is avoided. As long as a file is not shared, there is no communication over the network.

New file-version numbers are generated by a versioning server. If two different updates are made to the same file, the change with the higher version number will shadow the change with the lower version number. However, no change is lost. The system detects update conflicts and signals the user to invoke an application-specific resolver that retrieves conflicting versions and merges them to create a new version. When other nodes see a request for a new version number or its corresponding repair message, they can mark the corresponding file in their caches as stale. To deal with situations in which both messages are lost, JetFile maintains a Current Table that contains all the current file-version numbers for a particular volume. It has a limited lifetime, which limits the time a host may access stale data. When the network doesn't drop many packets, file consistency can be as good as in AFS. However, in the worst case, it will only be as good as in NFS.

Gronwall presented performance-measurement numbers using the Andrew benchmark. For the hot cache case, the performance of JetFile over a LAN is similar to that of a local-filesystem UFS. For the cold cache case, Gronwall compared the performance of JetFile over a LAN to its performance over an E-WAN with round-trip time of 0.5 seconds. The time for the CopyAll operation increases dramatically for the E-WAN case, because CopyAll requires synchronous communication.

The first questioner asked about security, and what trust relationship can be expected from filesystem peers. Gronwall answered that no trust is assumed between the hosts. As in other systems, files can carry signatures or be encrypted. The performance measurements indicate that there is rarely redundant communication. Therefore, the amount of work for encryption and verification should be similar to or less than that in most conventional designs.

The second question addressed the issue of limiting the number of filename-to-multicast-channel hashes that routers need to store. Gronwall suggested some possible ways of dealing with this. One is to limit the range of the hash function. What is perhaps more interesting is that you can use "wake-up messages" for volumes that are not referenced in a long time. State for such files need not be stored in routers until a wake-up message brings the servers back from idle status.

David Anderson of the University of Utah asked about JetFile's scalability in terms of the number of clients, because the SRM message volume will scale up quickly. Gronwall agreed that the system needs some form of locality. You can use IP multicast scope to make a smaller request first and only go beyond the local scope if necessary.

More information on the project is available at http://www.sics.se/cna/dist_app.html.

Integrating Content-based Access Mechanisms with Hierarchical File Systems

Burra Gopal, Microsoft Corporation; and Udi Manber, University of Arizona

Manber, who now works at Yahoo!, started by emphasizing that this work was done when both authors were at University of Arizona, and should not be seen as an indication of a mysterious collaboration between Microsoft and Yahoo!. Since both authors have moved on to other areas, Manber hopes that people will realize this is an important area and take over the work.

Manber claimed that one of the main challenges for operating systems in the future will be providing convenient access to vast amounts of information. Here the word "convenient" refers not only to speed, but also to the ability to find the right information. Filesystems today use the same paradigm as 30 years ago: a hierarchical naming system in which the user has to do the naming and remember the names. This paradigm

does not scale for gigabyte or terabyte systems, because users will not remember where everything is. Users should be able to easily access files with certain attributes, such as "anything I changed on Tuesday" or "anything with foo AND bar."

Gopal and Manber began with the Semantic File System (SFS) by Gifford et al. SFS builds virtual directories where the name of the directory corresponds to a query, and the content of the directory is files that match that query (represented by symbolic links).

Gopal and Manber combined SFS's features with a regular UNIX filesystem and allowed users to add semantic directories in addition to regular directories. Each semantic directory acts like a regular directory: users can add, remove, and modify files in it as usual. Query results can be modified automatically or manually. Regular file operations are preserved, while semantic-directory operations are added, such as `smkdir` (create a query), `smv` (change the query), `ssync` (reevaluate the query), and `smount` (define a semantic mount point).

When a user specifies the query associated with a semantic directory, the system puts symbolic links to all files matching the query into the directory. The symbolic links are called transient links because users can add or remove them. If a user adds a file, the link becomes permanent. If a user removes a file, the link becomes prohibited.

Manber next addressed the issue of scope consistency. Each query has a scope, and a subdirectory is a refinement to the query. For example, for semantic subdirectory `foo/bar`, `bar` evaluates only items within `foo`. When the user changes `foo` or moves `bar` to somewhere else, `bar` has to reevaluate the query because of the different context. Such changes can cause scope-consistency problems or even cycles of dependency. The paper presents a reasonable way of handling scope consistency. By design, the system does not

handle data consistency. Queries are evaluated only periodically or when instructed by the user, not all the time.

Another interesting idea is semantic mount points, which allow users to connect to different query systems (such as Yahoo!). This allows sharing not only of data but also of classifications or other ways the data are organized. The result is that users can treat the Web or other filesystems as part of their own directories.

Manber briefly talked about the implementation of the system. It was built on top of SunOS using the search engine Glimpse. It uses about 25,000 lines of C code. A major design decision was not to make kernel modifications so that people can easily accept the system. This most likely had a large impact on performance, with 30-50% time overhead and 10-15% space overhead for directory operations. In closing, Manber reiterated that the problem will not go away and will only become harder. He said this work has proven the approach is feasible, but not that it is the right approach. In particular, a user study would be needed.

David Steere of OGI referred to his SOSP paper on improving the performance of search engines. In that work, the results of queries were made immutable. He asked whether they had other ways to get around the possible usability problems caused by query mutations. Manber answered that while their design tried to be as "natural" as possible, he didn't really know what users will find to be "natural." This is a new issue, and a lot of work needs to be done, Manber said.

Steere then noted the similarity between Gopal and Manber's filesystem and a database. He asked if Manber thought filesystems will still be around 10 years from now, or if we will just be using databases. Manber said you can view filesystems as databases, and the question is how structured those databases will be. His guess is that there will be all kinds, and he said he hopes the dominant ones

will be less structured than the filesystems of today.

Works-in-Progress Session

Summaries by Xiaolan Zhang

Multi-Resource Lottery Scheduling in VINO

David Sullivan, Robert Haas, and Margo Seltzer, Harvard University

Lottery scheduling's ticket and currency abstractions can be used to manage multiple resources (CPU, memory, disk, etc.). Sullivan described extensions to the lottery-scheduling framework designed to increase its flexibility while preserving the insulation properties that currencies provide. Ticket exchanges allow applications to modify their resource allocations by trading resource-specific tickets with one another, and they do so without affecting the resource rights of nonparticipants. VINO's extensibility mechanism can be used to install resource negotiators that initiate exchanges; currency brokers that provide flexible access controls for currencies; and specialized, per-currency scheduling policies.

Quality of Service Support in the Eclipse Operating System

John Bruno, Jose Brustoloni, Eran Gabber, Banu Ozden and Avi Silberschatz, Bell Labs, Lucent Technologies

Brustoloni described Eclipse/BSD, a system derived from FreeBSD to provide the QoS support required by an increasing number of applications. Eclipse uses resource reservations to guarantee that a given client receives the QoS that it requests. Resource reservations enable hierarchical proportional sharing of all resources in the system. Using separate resource reservations, servers can guarantee that the requests of a given client are isolated from the influence of overloads caused by other clients. Applications specify resource reservations using a new /reserv filesystem API. Results show that Eclipse/BSD can improve the isolation

between Web sites hosted on the same system.

Long-Term File System Read Performance

Drew Roselli and Jeanna Neefe Matthews, University of California, Berkeley; Tom Anderson, University of Washington

Neefe Matthews described studies based on traces of long-term file behavior that show that, even with large caches, read performance is still significantly affected by disk seeks. They have therefore examined the impact of different layout policies on reads, including a new historically based policy that outperforms FFS and LFS on all workloads they have examined but requires many disk reorganizations. They are working on ways to limit the number of reorganizations required and to quantify their overhead.

High-Performance Distributed Objects over a System Area Network

Alessandro Forin, Galen Hunt, Li Li, and Yi-Min Wang, Microsoft Research

Wang described optimization techniques to improve DCOM performances over a system-area network with user-level networking. In particular, he and his colleagues are interested in the performance of distributed applications on top of the Virtual Interface Architecture (VIA). They applied both runtime and transport optimizations, and they removed an extra copy at the marshaling layer, yielding significant improvements in latency and throughput. Wang summarized by noting that fast networks push the bottleneck to protocol stacks, user-level networking pushes the bottleneck to the distributed infrastructure, and their optimization techniques push the bottleneck to transactions and security.

The Pebble Component-Based Operating System

Eran Gabber, John Bruno, Jose Brustoloni, Avi Silberschatz, and Christopher Small, Bell Labs, Lucent Technologies

Gabber presented the Pebble operating system, which allows system programmers to mix and match dynamically replaceable, user-level components. It includes a minimal, privileged-mode nucleus. IPC is done via portals, which are synthesized dynamically by a portal manager. Because each portal is specific to a particular caller and callee, it can be optimized to run fast. Each component is a protection domain and contains its own portals. Pebble is intended as a platform for high-end embedded applications.

Cellular Disco: Resource Management Using Virtual Clusters on Scalable Multiprocessors

Kinshuk Govil, Dan Teodosiu, Yongqiang Huang, and Mendel Rosenblum, Stanford University

Kinshuk Govil noted that system software that fully utilizes the features of large-scale multiprocessors is still not available, since most commercial operating systems do not provide efficient management of hardware resources or fault containment for such processors. Govil and his colleagues address this problem by running multiple instances of an off-the-shelf OS on top of a virtual machine monitor. The multiple OS instances talk to one another using a distributed-systems protocol and form a virtual cluster. A prototype implementation on an SGI Origin 2000 with 16 processors shows that faults can be isolated to a single OS instance and that the performance overhead is less than 10%.

PerDiS: A Persistent Distributed Store for Cooperative Engineering

Xavier Blondel, INRIA

Blondel described the PerDiS architecture, which was developed to support

data sharing by members of the construction industry involved in a common project. PerDiS provides a unique combination of features, including persistent shared memory, shared objects, security, and transparent persistence through a garbage-collection mechanism. The system is intended to be used in a large-scale WAN environment such as the Internet.

Resource Management in a Multi Computer System (MCS)

Dejan S. Milojicic and the MCS Team, Hewlett-Packard

Milojicic introduced MCS, a shared-memory machine running multiple copies of NT on multiprocessor nodes, and the mechanisms and policies needed to manage MCS resources. These mechanisms include global memory-management support and global schedulers for initiating processes on the nodes and scheduling I/O on the devices. Policies are used to make decisions based on resource usage. Innovations of the system include several simplifying assumptions that make the design and implementation of resource management easier, e.g., a limited single system image, distributed memory management based on hardware, and intelligent I/O processors.

ISTORE: Introspective Storage for Data-Intensive Network Service

Aaron Brown and David Oppenheimer, University of California, Berkeley

Oppenheimer described ISTORE, a hardware/software architecture that enables the rapid construction of self-monitoring, adaptive single-purpose systems. A system built using ISTORE couples LEGO-like plug-and-play hardware with application-specific, programmer-specified policies for "introspection" (continuous self-monitoring and adaptation) in the face of changes in workload and unexpected system events such as hardware failure. It can thereby provide high availability, performance, and scalability while reducing the cost and com-

plexity of administration. Adaptability is enabled by a combination of intelligent self-monitoring hardware components, a virtual database of system status and statistics, and a software toolkit that uses a domain-specific declarative language for specifying application-specific monitoring and adaptation policies.

SafeThreads: New Abstraction of Control and Protection

Masahiko Takahashi and Kenji Kono, University of Tokyo

SafeThreads, a mechanism that provides fine-grained protection domains for multiprocessor systems, allows threads to execute safely in the presence of malfunctioning external components. Takahashi described an efficient implementation of this mechanism based on "multi-protection" page tables that allow each virtual memory page to have multiple protection modes at the same time. At any moment, one of the protection modes is effective on each processor. Context switches involve a simple change of the effective protection mode without other high-latency operations such as TLB flushes. The implementation doesn't require special hardware support.

Fast and Predictable Automatic Memory Management for Operating Systems

Godmar Back, Jason Baker, Wilson Hsieh, Jay Lepreau, John McCorquodale, Sean McDirmid, Alastair Reid, and Joseph Zachary, University of Utah

Reid and his colleagues are writing significant parts of operating systems in modern languages such as Java. To improve the performance and predictability of such languages, the authors are developing techniques to "stack allocate" objects to avoid heap allocation and garbage collection. First, they are measuring object lifetimes through system tracing. In addition, they have developed a static analyzer that approximates the lifetimes of objects and determines at load time the activa-

tion record in which an object may be allocated.

File System Fingerprinting

Drew Roselli and Jeanna Neefe Matthews, University of California, Berkeley; Tom Anderson, University of Washington

Roselli pointed out that current filesystem implementors face the following dilemma in layout decisions: the filesystem has more information about the likely access patterns of files, but the storage system has more information about the performance of the storage media. She proposes an enriched filesystem/storage system interface that allows the filesystem to provide abstract rather than absolute file positions to the storage system. The enriched interface can improve performance in the following way: For predicted next write, the storage system can retain the data in write buffer; for predicted next read, the storage system can perform read-optimized layout. The filesystem can also provide relative block placement information that infers abstract data relationships.

Agile: A Hierarchically Extensible Security Policy Architecture

Dave Anderson, Ray Spencer, Mike Hibler, and Jay Lepreau, University of Utah

Andersen observed that contemporary diverse operating environments call for hierarchically extensible and flexible security policies. Agile is a security-policy architecture that provides policy-independent, hierarchical extensibility. Agile borrows techniques from network-packet routing. In particular, it uses integer-namespace routing. Children are assigned SIDs and parents route decisions to their children using a routing algorithm such as the Patricia algorithm.

SAGEnews & features

Where's the Horse?



by Tina Darmohray

Tina Darmohray, editor of SAGE News & Features, is a consultant in the area of Internet firewalls and network connections and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tmd@usenix.org>

Those of you who know me well enough to know what I waste money on know that I've been sponsoring (as my husband refers to it) horses for as long as I've been financially able. I think horses are great, and I like just about everything to do with them: brushing, feeding, showing, and of course, riding. As far as basic transportation is concerned, I think riding a horse out along the trails is a fantastic way to get around. Of course, it's not really practical to use horses as transportation in our society, but I'd be all for it if it were put to a vote. Sadly, I can't say I feel the same way about certification of system administrators; at least not as I've seen it approached so far.

The certification debate has been going on for longer than I'd like to remember. The handful of people involved in originally organizing SAGE can attest that

even then it was one of the polarizing topics we discussed. Those for certification asserted it would benefit system administrators and the system-administration profession. Those opposed had a fundamental concern that certification might actually hurt the status of the profession. Since "advancing system administration as a profession" is at the core of the SAGE mission, it's not hard to understand why a topic that produces exactly opposing opinions about whether it will help or hurt that goal is still so hotly debated. In close to a decade I have seen no change surrounding the certification issue, including the arguments for and against, and the lack of consensus among those in the profession.

Despite the status quo nonconsensus, the guild is pursuing certification through a certification committee, advisory council, and professionals in the field of certification program and test development. Given that, it's imperative that the resulting SAGE certification help the lot of system administration rather than hurt it. The key to doing so is for certification to convey, without exception, that system administration is a profession. Anything else is a giant step backward and will only undermine the good work that's already been done.

The reason that promoting professionalism is the core issue in certification, and in everything that SAGE pursues, is the

historical job misclassification that system administrators are trying to overcome to improve their careers. Remember that, until recently, "system administrator" was not a job title in many organizations; they had either operators (technicians who babysat mainframes) or computer programmers (degreed professionals who cut code). Those who found themselves misclassified as operators typically received less respect and less pay, while those misclassified as programmers received poor reviews because they didn't produce as much code as their programming peers. The goal of SAGE was to unite system administrators in an effort to classify system administrators correctly as degreed computer professionals that didn't cut code for a living, and to provide a credible platform from which to launch that effort.

System administration is not the first profession to struggle with professional nomenclature. Historically, when new technology creates new jobs, the traditional professional trappings, such as job classifications, degrees, organizations, and certifications, lag behind. System administration is no different. Since the demand for the work exists, you find individuals qualifying for the positions through a variety of degree-equivalent, and on-the-job-training, experience. Over time, formal education catches up,

SAGE, the System Administrators Guild, is a Special Technical Group within USENIX. It is organized to advance the status of computer system administration as a profession, establish standards of professional excellence and recognize those who attain them, develop guidelines for improving the technical and managerial capabilities of members of the profession, and promote activities that advance the state of the art or the community.

All system administrators benefit from the advancement and growing credibility of the profession. Joining SAGE allows individuals and organizations to contribute to the community of system administrators and the professions as a whole.

SAGE membership includes USENIX membership. SAGE members receive all USENIX member benefits plus others exclusive to SAGE.

SAGE members save when registering for USENIX conferences and conferences co-sponsored by SAGE.

SAGE publishes a series of practical booklets. SAGE members receive a free copy of each booklet published during their membership term.

SAGE sponsors an annual survey of sysadmin salaries collated with job responsibilities. Results are available to members online.

The SAGE Web site offers a members-only Jobs-Offered and Positions-Sought Job Center.

SAGE STG EXECUTIVE COMMITTEE

President:

Hal Miller <halm@usenix.org>

Vice-President:

Barbara L. Dijkster <barb@usenix.org>

Secretary:

Tim Gassaway <gassaway@usenix.org>

Treasurer:

Peg Schafer <peg@usenix.org>

Members:

Xev Gittler <xev@usenix.org>

Geoff Halprin <geoff@usenix.org>

Jim Hickstein <jim@usenix.org>

and the community of professionals has a more homogeneous, and formal, educational background. The problem for system administrators has been that during this transition many were misclassified as technicians, which hurt the overall status of system administration as a profession. Luckily, the formation of SAGE and the publication of the SAGE *Job Descriptions* booklet have helped with the lack-of-formal-job-classifications aspect.

It seems that the next logical step at this crossroads for system administration would be outlining and creating avenues for formal education. As a profession we need to provide the educational support, curriculum requirements, course outlines, textbooks, etc., to ensure that degrees in our field (or majors with an emphasis on system administration) are available. The recently published *Educating and Training System Administrators* SAGE booklet is a good start down this path, but now SAGE needs to insure that those good ideas are put in place so that formal system-administration courses and degrees are available. Indeed, these educational credentials are really the first "certification" that is needed and the vehicle for any others that would follow.

After formal educational guidelines, certification may also be desirable. For instance, engineers have had this kind of "certification" in place for years, in the

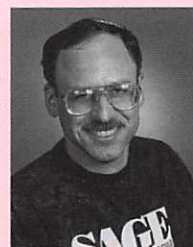
form of "Professional Engineer." Out of curiosity, I called the California State Board for Professional Engineers, and here's what I found out:

A degree is not required to be a Professional Engineer; however, every PE needs to pass the Engineer-in-Training (EIT) exam (the baseline test of the "fundamentals of engineering," roughly equivalent to three years of college engineering education/three years of engineering work experience). Once you pass the EIT, you can take the PE test for a particular branch of engineering. (I asked about nuclear engineering, since that's what my father is in.) For that exam you must submit an application, including transcripts and references, demonstrating that you have six years of applicable education/experience. The application runs through three levels of "verification" (i.e., it's not the honor system). If your application checks out, you get to take the exam to be a Professional Nuclear Engineer. If you pass, you are one!

I'm anxious to hear what the "professionals in the field of certification, program, and test development" who are working with the SAGE certification subcommittee and advisory council suggest. I'm hoping that they follow the model of Professional Engineers, positioning certification as an affirmation of experience, rather than a shortcut to bypass education and training. I suppose, like horses,

education and training are my vehicles of choice for advancing our profession, and creating certification prior to developing the educational infrastructure sure feels like putting the cart before that horse.

Note from the President



by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

<halm@usenix.org>

A year and a half ago I ran a survey. While it was primarily aimed at the certification question, it had a number of other issues attached. The one that drew the most clear and positive response (overwhelming, in fact), was my "How-To Notes" series. We heard you, and it is now into production. Hal Pomeranz jumped right in and wrote the first one as a test case, and that has already graced the pages of this publication. A second one, by Adam Donahue on Apache, is in this issue.

After a series of negotiations (things

SAGE MEMBERSHIP

<office@usenix.org>

SAGE ONLINE SERVICES

Email server: <majordomo@usenix.org>

Web: <<http://www.usenix.org/sage/>>

SAGE SUPPORTING MEMBERS

Atlantic Systems Group

Collective Technologies

Deer Run Associates

D.E. Shaw & Co.

Electric Lightwave, Inc.

ESM Services, Inc.

Gnac, Inc.

Mentor Graphics Corp.

Microsoft Research

MindSource Software Engineers

Motorola Australia Software Centre

New Riders Press

O'Reilly & Associates

Remedy Corporation

SysAdmin Magazine

Taos Mountain

TransQuest Technologies, Inc.

UNIX Guru Universe (UGU)

move slowly in volunteer organizations), we now have an editor for the series. Melissa Binde will be shepherding, cajoling, recruiting, and chasing up as necessary to build this into a large, living collection. Her first effort (which we hope will be completed by the time this reaches your mailbox) will be to finalize the processes of the series and formats of the Notes. Next comes Web page production. She will then, if I know her as I think I do, come knocking on your virtual door looking for ideas and authors, so get your list of suggestions ready!

The other big winner in my survey was the question about involvement in the standards process. I can use some additional help on this one. Nick Stoughton represents USENIX and SAGE to the various standards bodies (and I can say from personal observation, he does it well). As to system-administration standards, though, the one area that sounded promising for our involvement did not pan out. There appears to be very little going on in sysadmin standardization. We could use some ideas on where to get involved. I'd like to see us select, if possible, some areas where we might have some positive impact and something solid to offer. Please email Nick <nick@usenix.org> and/or me if you have any thoughts on this.

One reason we may not (yet) have the level of influence we may want in the standards process is our size. While forming the organization and defining the profession, we haven't worried much about our penetration of the potential membership market. We are ready now to begin a significant expansion. We've geared up by creating a vice president role to further split duties, creating a deputy executive director position that works much of the time on SAGE issues, etc. Now it's time to find the people who "ought" to be members. Certainly every one of us should put some effort into recruiting, but even if each of us brings two more, we still will have only a couple

of percent of the sysadmin population. The USENIX marketing director is doing some additional publicity work. We need yet more and better ways to reach the target. We have begun working with vendors to put a SAGE brochure into the box when they ship a computer (figuring that it is a "sysadmin" who opens it). Other ideas are solicited. Send email to Cynthia Deno <cynthia@usenix.org> and/or me.

SAGE is what all of us make it. Here are a couple of simple, quick things you can do, just by emailing ideas. I hope to hear from you!

Preparing to Consider Certification



by Bryan McDonald

Bryan McDonald is a program manager at GNAC, where he leads consulting teams on systems, networks, and security projects for a variety of customers.

<bignac@gnac.com>

SAGE had many goals in the early days, probably as many as there were people interested in participating in its formation. Even then, certification evoked the most passion in us all, both those for and those against it. It seemed so right that an organization founded to advance the "profession" of system administration should take this issue on and do it "now." Then, as now, lots of vendors were already offering certification courses, which fueled the sense of urgency.

Unfortunately, the fact that certification courses were springing up in various vendor arenas, and even in training schools, has hurt the certification debate rather than clarified it. It is easy to react to their presence, to dismiss the bad ones, yet

mistrust the trend and feel anxious about its impact on our jobs. It is easy to decide that SAGE needs to drive a better program, one that truly defines who we are and the value we bring to our employers and communities. It is hard, however, to define the value. Many arguments involve the very core definitions of what we do: Is designing ATM networks the same as installing user accounts? Is editing the registry the same as editing the `resolv.conf` file? Is managing a few machines for some Ph.D.s in a faraday cage the same as managing the backbone for a 100,000-node network? In many ways, we aren't even sure yet what it is we are certifying, so how can we certify it?

In the early days of medicine, doctors cared for small communities of people, learning about their strengths and weaknesses, understanding the foods, the environment, and the hardships of their lives. They cared for the whole community, from birth to death, and they trained each following generation as best they could, passing on a bit more information than their teachers had. Eventually the communities got larger, the cures got more complicated, and the village doctors began learning about the medicines and cures from other villages. When the task of passing on this knowledge got to be too great for one person to accomplish, the doctors gathered and formed organizations dedicated to learning more and to teaching more to the next generation. Long before the AMA came into existence, schools and universities formed that taught young doctors how to heal and gave them common ground so that they understood one another. Certifying the skills and principles of healing – authenticating the study, the learning, the experience – was a logical next step.

System administration is not dealing with life-and-death issues (most of the time), but the complexity of the task before us is not all that dissimilar either. How can we even begin to codify the standards

and practices that make up the multiplicity of things that we all do, until we first develop a framework for what we do? How can we certify that someone is versed and experienced in that framework until we teach it to them? How can we certify until we educate?

Think about the vendors out there. Each of them wants to certify that you know how to do something, something of value to an organization. They offer training courses, tutorials, and other events for you to learn more about that "something." Then they certify that you have experienced their educational program. SAGE cannot begin to consider certifying a system administrator until the framework is taught. Once that is accomplished, then we will be prepared to consider certifying that people know this framework.

Can we define the framework? Yes. Have we? Not yet. But we can. And we should. We should help build educational resources that can define the framework from vendor schools to universities. In this way we can define the value of our learning and experience, and truly advance the state of the profession

SAGE Certification Update



by Gale Berkowitz

USENIX Deputy Executive Director

<gale@usenix.org>

The SAGE Certification Project is well underway. We are very excited about this effort because it will help to establish industry standards to better prepare and train professionals in the field.

The certification project has four phases:

Phase 1: Program planning

Phase 2: Occupational analysis and assessment design

Phase 3: Development of strategies for influencing education and training activities

Phase 4: Development and implementation of the certification program

Phase 1 is being used to solidify project goals, plans, and procedures associated with the project as a whole. Analysis of the competencies required of systems administration professionals has begun.

As part of the first steps of the systematic occupational analysis, several focus groups are being conducted over the next few months that draw from a broad cross-section of professionals in the field. Input is being sought about the tasks and responsibilities that comprise the job at various levels of experience. We will also be determining the key knowledge, skills, and abilities that are required. The focus groups are being led by a representative of the Human Resources Research Organization, or HumRRO, the contractor firm that has been hired to help us conduct the occupational analysis.

The second phase, the occupational analysis, will use available materials, incumbent interviews and workshops, and a survey of job incumbents to describe the core requirements of the system administrator occupation. Information derived from the focus groups will be used to develop the automated survey of systems administrators. The information derived from the occupational analysis will be used to determine the scope and content of Phases 3 and 4.

We are also seeking support from outside organizations to both fund and fuel this endeavor. While our project, and products, will be vendor-neutral, we expect that much of the information derived from this project will be useful for commercial applications. If you are interested in sponsoring this effort, please contact Gale Berkowitz at <gale@usenix.org>.

applying sendmail anti-spam checks at the user level



by Bailey Szeto

Bailey Szeto is a systems administrator for Cisco Systems Inc. When not dealing with mail issues, he enjoys . . . oh wait, there's never a time when he's not dealing with mail issues. . . .

<beetle@cisco.com>

For most corporations, email is a mission-critical application. It often is the number one communications medium for developers, sales, and customers. However, unsolicited commercial email (UCE or spam) has reached levels at which it is starting to interfere with the effectiveness of email as a communication tool. Separating junk from real email wastes not only network/computing resources but also employee time. More important, many people consider spam to be an invasion of their private mailboxes; arguably the worst aspect of spam is that it demoralizes employees and can even jeopardize their emotional well-being.

Most corporate postmasters have been given the responsibility of dealing with spam. A quick search on the Internet reveals various technical solutions that have been created to help stop spam. One big implementation problem with these anti-spam measures is that they are usually applied on a site-wide basis. For most corporations, some email addresses – such as sales, technical support, and bug reporting – *must* not be blocked. Some of the spammers are our customers; we want their purchase orders to get through but not their spam. We never want to block bug reports from coming in, even if they are from a known spammer.

This article discusses configuration changes that can be made to sendmail rulesets in order to implement an anti-spam filtering policy on a per-user basis. Users can decide if they want to activate anti-spam features and what level of filtering they want.

The Anti-Spam Features of sendmail

Beginning with sendmail 8.8, the `check_*` group of rulesets were added as features. This group of rulesets provides hooks into the SMTP dialog. For the sake of clarity, I'll show the SMTP dialog here:

1. The sending machine issues a `HELO` (or `EHLO`) in which it identifies itself.
2. The sending machine issues a `MAIL FROM` in which it identifies the sender of the message.
3. The sending machine issues a `RCPT TO` in which it identifies the recipient of the message.
4. The sending machine issues a `DATA` to tell the receiving machine it is about to transfer the message.
5. The message is transferred, and the sending machine ends the message with a `."` on a line by itself.
6. The receiving machine acknowledges that it got the message, usually by issuing a unique number.

Sendmail 8.8 included the following four check rulesets:

- `check_relay` - this ruleset is called after step 1 in the SMTP dialog above. It is used to prevent unauthorized IPs from connecting to your machine.

- `check_mail` - this ruleset is called after step 2 in the SMTP dialog. It is used to stop mail from known senders.
- `check_rcpt` - this ruleset is called after step 3 in the SMTP dialog. It is primarily used to stop relaying (not to be confused with `check_relay` above.) Relaying occurs when an external user sends mail to your server meant for a different external user. They are using your server as a relay for their email. Spammers often do this in order to hide their identity or to take advantage of your resources. Since we know both the sender and recipient at this point, we can decide whether or not the email is relayed.
- `check_compat` - this ruleset is called after step 5 in the SMTP dialog. It can be used to stop delivery of a message after it has been accepted.

Although these `check_*` hooks were provided, it was left to the system administrator to actually develop rules using these hooks. Claus Assmann[1] and Robert Harker[2] maintain a set of effective rules based on these hooks.

When Sendmail 8.9 was released, Eric Allman included some basic anti-spam features that could be configured into sendmail to take advantage of these hooks. By default, Sendmail 8.9 had relaying turned off (implemented in the `check_rcpt` ruleset). Furthermore, you could enable rejection of email based on either a DNS lookup or the results of a database lookup (implemented in the `check_mail` ruleset).

The Problem

The main problem with the anti-spam features included with sendmail is that the checks are made too early in the SMTP dialog. As configured by sendmail, both the DNS and database check are made in `check_mail` (SMTP step 2), after the sender has been identified. If the sender fails the checks, the mail is rejected.

The rejection comes too early because we do not know whom the mail is meant for yet. Also, this means that mail will be bounced regardless of who the recipient was. This is a problem for corporations because there may be some addresses that *must* receive all email. Also, some users may actually want to get spam (true case)!

The Solution

I thought about ways we could block spam for our users while at the same time allowing full access for other addresses. After a little experimenting, I came up with a ruleset that I call "Extended_check_rcpt." Basically, I hold off on the spam checks until the recipient is identified. Then we can check to see if the recipient wants filtering and apply the spam checks as appropriate.

At first I thought about implementing this delayed check by taking advantage of `check_compat`. According to the sendmail book, "Not all situations can be resolved by simply checking the recipient or sender address. Sometimes you will need to make judgments based on pairs of addresses. To handle this situation, V8.8 introduced the `check_compat` rule set." [3] Unfortunately, the problem with `check_compat` is that it is called after the message has been accepted. That means that the sender has already transmitted the message and has closed the connection. If you decide to bounce the message because it fits the spam criteria, your server is then tasked with delivering a bounce message back to the sender. If the sender's address is fake, the bounce messages may back up and clog your mail queue.

Ideally you want to be able to reject a message before it is accepted and the sender has closed the connection. This will shift the burden of delivering a bounce message back to the sending machine. Therefore the best place to apply our spam checks is in

The main problem with the anti-spam features included with sendmail is that the checks are made too early in the SMTP dialog. . . . Mail will be bounced regardless of who the recipient was.

Since sendmail supports the use of a database to keep track of spamming addresses, we can create another database to keep track of user preferences.

`check_rcpt`, after both the sender and recipient are identified but before the message is sent. Fortunately, sendmail stores the sender's address in a macro, and we can use sendmail's delayed macro expansion capabilities to access this value during `check_rcpt`.

Since sendmail supports the use of a database to keep track of spamming addresses, we can create another database to keep track of user preferences. After our modifications are done, the SMTP dialog would look something like this:

1. The sending machine issues a `HELO` (or `EHLO`) in which it identifies itself.
2. The sending machine issues a `MAIL FROM` in which it identifies the sender of the message.
3. The sending machine issues a `RCPT TO` in which it identifies the recipient of the message.
 - a. Look into the user database to see if the recipient wants spam filtering.
 - b. Apply DNS check if appropriate.
 - c. Apply spam database check if appropriate.
 - d. Reject the message if step b or c fails, otherwise continue with step 4.
4. The sending machine issues a `DATA` to tell the receiving machine it is about to transfer the message.
5. The message is transferred, and the sending machine ends the message with a `.` on a line by itself.
6. The receiving machine acknowledges that it got the message, usually by issuing a unique number.

The new ruleset is called `Extended_check_rcpt` because it is called after the sendmail's `Basic_check_rcpt`, which in turn is called by `check_rcpt`.

sendmail.cf Changes to Implement Extended_check_rcpt

I'll assume that you already know how to create a `sendmail.cf` file from an `mc` file. You can have other features in your `mc` file, but the two you should have in order to implement `Extended_check_rcpt` are:

```
FEATURE(access_db, dbm -o /etc/mail/access)dnl
FEATURE(accept_unresolvable_domains)dnl
```

Even though the access database checks too early during the SMTP process, it is still a useful feature to enable because the database serves other purposes. First, it is used to enable selective relaying. By listing domains in the access database, you can allow other domains to relay through your site. Second, there may be instances in which you really want to block email from a particular domain, regardless of users' settings. You can use the access database to globally block a particular sender or domain. However, we will not use the access database for "spam stomping," as the config file puts it. We will put our spammers into a different database.

The second feature, "accept unresolvable domains," is necessary because by default sendmail will block email coming from domains that do not exist. As with the access database, this check comes too early. We need to disable this feature so that the DNS check doesn't get included in the config file at `check_mail`. Instead, we will use our own custom code in `check_rcpt` to do the DNS check.

Once the `cf` file has been generated, you will need to hand-edit it to make a few changes.

1. Define the databases. A good place to add the database definitions is after the access database line. My additions are in bold:

```
# Access list database (for spam stomping)
Kaccess dbm -o /etc/mail_access

# Spam database (database of known spammers)
Kspammer dbm -o /etc/spammer

# User opt-in database
Kspamuser dbm -o /etc/spam_user

# Resolve map
Kresolve host -a<OK> -T<TEMP>
```

Explanation: The `K` configuration options tell sendmail that we will be using two databases (referenced by the names `spammers` and `spamuser`). The databases are set up as `dbm` files, but you can substitute whatever database format you are comfortable with here (`db`, `hash`, etc.). The `Kresolve` map is needed for the DNS check.

2. Modify the current `check_rcpt`. The `check_rcpt` ruleset as created by sendmail will look like this (note that the numbers on the left are for reference only and will not appear in the `cf` file itself):

```
1 Slocal_check_rcpt
2 Scheck_rcpt
3 R$*          $: $1 $| $>"Local_check_rcpt" $1
4 R$* $| $#$*  $#$2
5 R$* $| $*    $@ $>"Basic_check_rcpt" $1
```

Change the `check_rcpt` ruleset so that it reads:

```
1 Slocal_check_rcpt
2 Scheck_rcpt
3 R$*          $: $1 $| $>"Local_check_rcpt" $1
4 R$* $| $#$*  $#$2
5 R$* $| $*    $: $1 $| $>"Basic_check_rcpt" $1
6 R$* $| OK    $: $1 $| $>"Extended_check_rcpt" $1
7 R$* $| $*    $: $2
```

Explanation: The first thing needed is to change line 5. You'll see that I replaced the `$@` with a `$:`. The reason is that `$@` tells sendmail to do the rewrite and exit the rule set. Instead, we want sendmail to do the rewrite and continue to the next line in which we call our customized checks. Also, note that I prepended a `$1 $|` before the command to execute `Basic_check_rcpt`. This is needed because `Basic_check_rcpt` is called with the recipient name but returns with a status condition (`OK`, `err`, etc.). We still need to keep track of the recipient so that we can call our customized rulesets with it.

As an example, let's say that the recipient is `bond@martini.com`. In the original unmodified ruleset, line 5 would be called with the argument `bond@martini.com`. However, the entire workspace is replaced with the results from `Basic_check_rcpt`. Assuming that `Basic_check_rcpt` finds the address acceptable, the workspace would read just `"OK"`. We have lost the original address and cannot use it to call our customized ruleset.

However, with our modified ruleset, line 5 would rewrite our workspace to be:

```
bond@martini.com $| (whatever Basic_check_rcpt returns).
```

Assuming `Basic_check_rcpt` finds the address to be acceptable, our workspace would now read:

You can use the access database to globally block a particular sender or domain. However, we will not use the access database for "spam stomping," as the config file puts it. We will put our spammers into a different database.


```
bond@martini.com $! OK
```

We can now check for the OK as the second token and call our customized ruleset `Extended_check_rcpt` with the original address that was saved as the first token. This is what happens with line 6.

3. Insert into your `cf` file the code for `Extended_check_rcpt`. This code should be inserted after the last line in `Basic_check_rcpt` but before the first line of mailer definitions. The source for `Extended_check_mail` is in the Appendix. You can find an online version of `Extended_check_rcpt` at <http://www.employees.org/beetle/sendmail.html>.

```
R$*$: [ $! ]put brackets around it...
R$=w    $@ OK                        ... and see if it is local
```

```
# anything else is bogus
R$*      $#error $@ 5.7.1 $: "550 Relaying denied"
[Insert code for Extended_check_rcpt here]
```

Mailer Definitions

The code for `Extended_check_mail` is basically the same DNS check code and access database code as is shipped with `sendmail`. I have broken out the checks into separate rulesets. I have also added some logic to check the user choice database, which then calls the separate rulesets as necessary. I'll comment on the interesting portions of

`Extended_check_mail`:

```
SExtended_check_rcpt
R$*$: $1 $! $>3 $&f
R$* $! $*$: $2 $! $>3 $1
R$* <@ $* > $! $*      $: <$1 @ $2> $! $3
R$* $! $+ < @ $* > $*   $: $1 $! $2 < @ $3 > $4 $! $2
```

Again, we are making use of the `$!` token to separate fields in our workspace. The code above basically rearranges our workspace so that it contains:

```
sender  $! recipient                $!      username
```

The username is just the recipient with the `@domain` chopped off. We will use the username to look into our database to see if this user wants spam filtering.

```
R$* $! $* $! $*   $: $1 $! $2 $! $(spamuser $3 $:<?NOKEY> $)

# No such user in database. Don't do any checks
R$* $! $* $! <?NOKEY>  $@ $2
```

After we get the username as the third field, we pass that into the database lookup. The `$:<?NOKEY>` tells `sendmail` to return `<?NOKEY>` as a default value if no entry is found. If `<?NOKEY>` is returned, the very next line tells `sendmail` to return the second field (the recipient) and exit the ruleset.

If the database lookup did return a value, then we will check that value and call `My_check_domain` (which does the DNS check), or `My_check_db` (which checks the spammer database), or both. Depending on the results of `My_check_db` or `My_check_domain`, we will return either the original recipient or an error and exit.

Creating the Databases

Once you are done making the `sendmail` configuration file changes, you'll need to create the databases that contain the information. The `sendmail` distribution comes with a handy tool to do this, called `makemap`. My sample entries are shown in bold. We'll use these sample entries for testing later.

1. The `spam_user` database is where your users' preferences are kept. It will be used to tell sendmail if the recipient wants DNS checking, spam database checking, or both. To create this database, first create a text file called `spam_user` in this format:

```
Username <choice>
```

where `<choice>` can be `<?DOMAIN>` for DNS checking, `<?DB>` for database checking, or `<?BOTH>` for both checks. After the text file is created, use `makemap` to create the database:

```
tim    <?BOTH>
brad   <?DOMAIN>
john   <?DB>
```

```
% makemap dbm /etc/spam_user < /etc/spam_user
```

Note that the location of the file (in my case `/etc/spam_user`) should correspond with the location you defined with the `Kspamuser` configuration command in step 1. Also, be sure that the database type corresponds.

2. Create the `spammer` database. This is the database where we will keep all of our known spammers. The procedure is essentially the same as creating the `spam_user` database. Make a text file called `spammers` that contains:

```
Address Message
```

where `Address` can be a fully qualified address (`somewhere@somewhere.com`), an address alone (`free.stealth.mailer@`), or just a domain (`somewhere.com`). The message can be either `REJECT` or a customized message (550 - We don't want spammers here).

```
free.stealth.mailer@      550 - We don't want spammers here
hot999@aol.com           REJECT
spamrus.com REJECT
```

```
% makemap dbm /etc/spammers < /etc/spammers
```

3. Create the access database. If you recall, we will not be using the access database for spam stomping but have configured it to take advantage of its other functions. This is the database where we will keep addresses that are allowed to relay and also addresses that will be globally blocked, regardless of how the users have their spam preferences set. Read the sendmail documentation on how to use the access database if you need to enable relaying. For the purposes of this article, I will create an empty access database:

```
% touch /etc/access
% makemap dbm /etc/mail_access < /etc/mail_access
```

Testing the New Configuration

It's very important to test the new configuration before using it in production. You can test the changes with sendmail's address-test mode (`sendmail -bt`).

Before testing, you will need to define the `f` macro value that holds the sender's address:

```
.Df<sender address>
```

Test Case #1 – Tim has enabled both DNS and database checking.

- A. Testing mail from someone in the spammer database to Tim. Since Tim has both checks turned on, the system should reject the email.

```
% /usr/lib/sendmail -bt
```

The `spam_user` database is where your users' preferences are kept. It will be used to tell sendmail if the recipient wants DNS checking, spam database checking, or both.


```
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
```

```
Enter <ruleset> <address>
```

```
> .Dfhot999@aol.com
```

```
> check_rcpt tim@cisco.com
```

```
[lots of output deleted]
```

```
rewrite: ruleset 3 returns: hot999 < @ aol . com . >
```

```
rewrite: ruleset 199 input: hot999 < @ aol . com . >
```

```
rewrite: ruleset 199 returns: hot999 < @ aol . com . >
```

```
rewrite: ruleset 183 returns: $# error $@ 5 . 7 . 1 $: "550 Access denied"
```

- B. Testing mail from a bogus domain to Tim. Since Tim has both checks turned on, the system should reject the email.

```
% /usr/lib/sendmail -bt
```

```
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
```

```
Enter <ruleset> <address>
```

```
> .Dfspam@garbagedomain.com
```

```
> check_rcpt tim@cisco.com
```

```
[lots of output deleted]
```

```
rewrite: ruleset 3 returns: spam < @ garbagedomain . com >
```

```
rewrite: ruleset 199 input: spam < @ garbagedomain . com >
```

```
rewrite: ruleset 199 returns: spam < @ garbagedomain . com >
```

```
rewrite: ruleset 180 returns: < PERM >
```

```
rewrite: ruleset 181 returns: < PERM >
```

```
rewrite: ruleset 184 returns: $# error $@ 5 . 1 . 8 $: "501 Sender domain must exist"
```

- C. Testing mail from a valid address not in spammer database to Tim. The sender address comes from a valid domain and is not in the spammer database, so the system should allow the email to pass.

```
% /usr/lib/sendmail -bt
```

```
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
```

```
Enter <ruleset> <address>
```

```
> .Dfuser@cisco.com
```

```
> check_rcpt tim@cisco.com
```

```
[lots of output deleted]
```

```
rewrite: ruleset 3 returns: tim < @ cisco . com . >
```

```
rewrite: ruleset 199 input: tim < @ cisco . com . >
```

```
rewrite: ruleset 199 returns: tim < @ cisco . com . >
```

```
rewrite: ruleset 179 input: < cisco . com > < ? > < >
```

```
rewrite: ruleset 196 input: < com > < ? > < >
```

```
rewrite: ruleset 196 returns: < ? > < >
```

```
rewrite: ruleset 179 returns: < ? > < >
```

```
rewrite: ruleset 183 returns: < OK >
```

Test Case #2 – Brad has only enabled the DNS check.

- A. Testing mail from someone in the spammer database to Brad. Since Brad has only enabled the DNS check, the mail should be accepted.

```
% /usr/lib/sendmail -bt
```

```
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
```

```
Enter <ruleset> <address>
```

```
> .Dfhot999@aol.com
```

```
> check_rcpt brad@cisco.com
```

```
[lots of output deleted]
```

```
rewrite: ruleset 3 returns: hot999 < @ aol . com . >
```

```
rewrite: ruleset 199 input: hot999 < @ aol . com . >
```

```
rewrite: ruleset 199 returns: hot999 < @ aol . com . >
```

```
rewrite: ruleset 180 returns: < OK >
```


- B. Testing mail from a bogus domain to Brad. Since Brad has enabled DNS checking, the system should reject the email.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfspam@garbage.com
> check_rcpt brad@cisco.com

[lots of output deleted]

rewrite: ruleset 3 returns: spam < @ garbage.com >
rewrite: ruleset 199 input: spam < @ garbage.com >
rewrite: ruleset 199 returns: spam < @ garbage.com >
rewrite: ruleset 180 returns: < PERM >
rewrite: ruleset 181 returns: < PERM >
rewrite: ruleset 184 returns: $# error $@ 5 . 1 . 8 $: "501 Sender domain must exist"
```

- C. Testing mail from a valid address not in spammer database to Brad. The sender address comes from a valid domain and is not in the spammer database, so the system should allow the email to pass.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfuser@cisco.com
> check_rcpt brad@cisco.com

[lots of output deleted]

rewrite: ruleset 3 returns: brad < @ cisco . com . >
rewrite: ruleset 199 input: brad < @ cisco . com . >
rewrite: ruleset 199 returns: brad < @ cisco . com . >
rewrite: ruleset 180 returns: < OK >
```

Test Case #3 – the undefined user.

I'll leave it as an exercise for the reader to try John's settings. One more test we should try is for the person who has not set up his or her spam settings. In this case, the system should allow all email to pass through.

- A. Mail from an address in the spammer database to Sally. Since Sally is not defined in the spam_user database, the email should be allowed through.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfhot999@aol.com
> check_rcpt sally@cisco.com

rewrite: ruleset 3 returns: sally < @ cisco . com . >
rewrite: ruleset 184 returns: sally < @ cisco . com . >
rewrite: ruleset 186 returns: sally < @ cisco . com . >
```

- B. Mail from a bogus domain should be ok too:

```
> .Dfspam@garbage.com
> check_rcpt sally@myhost.com

rewrite: ruleset 3 returns: sally < @ cisco . com . >
rewrite: ruleset 184 returns: sally < @ cisco . com . >
rewrite: ruleset 186 returns: sally < @ cisco . com . >
```

- C. Of course, mail from a valid domain that is not in the spammer database should be allowed through:

```
> .Dfbailey@cisco.com
> check_rcpt sally@myhost.com
```

We have created a Web page so that the user can log in, read about how we fight spam, and change the settings. The CGI then goes and updates the spam_user database.

```
rewrite: ruleset 3 returns: sally < @ cisco . com . >
rewrite: ruleset 184 returns: sally < @ cisco . com . >
rewrite: ruleset 186 returns: sally < @ cisco . com . >
```

Implementation Details

Now that you have tested the sendmail configuration changes, you will have to create some infrastructure to support the new features of `Extended_check_rcpt`. First, the user must have a way to change his or her settings in the `spam_user` database. We have created a Web page so that the user can log in, read about how we fight spam, and change the settings. The CGI then goes and updates the `spam_user` database. If you are a small site, you could probably get away with `rdisting` a text file and running `makemap` once in a while.

I think it is important to explain to the user the ramifications of turning on the spam checks. In the Web page that we use, we explain that the DNS check may be dangerous and that you should *not* turn it on if it is critical that you get all email. We even have a JavaScript popup that makes the user acknowledge that they have chosen a spam configuration that may cause them to miss email from misconfigured domains.

If the DNS check is too aggressive, then the user can just enable the spam database check. This option is pretty safe to use, since we will only enter addresses from known spammers.

We have set up a mailing list called `spam-fighters` for our employees to forward spam to. A volunteer team then looks at the email and adds the address to the spammer database if appropriate. Instead of telling users to “just delete the spam,” we now tell them to forward it to the `spam-fighters` alias so that the address can be blocked. Sometimes we can get a report in fast enough that we can block the address before the bulk of the spam comes through. Furthermore, we have created Web pages that show the daily spam statistics, who is in our spammer database, and the number of times we have blocked them. A user can submit a spam report, see that the address gets added to the database, and see the results of that report – that the spammer was blocked from further attempts at connecting to our servers. This is a big PR win for the sysadmin team and a psychological win for the user.

Finally, we try to be proactive about adding spammers into our spam database. We monitor `<news.admin.net-abuse.sightings>` for spam reports from other sites. We also have set up some fake addresses used by the `spam-fighters` team. We have publicized these fake addresses by posting to USENET news and using other methods. Since these addresses are not used by anyone inside of the company, any mail going to them is unsolicited and therefore classified as spam.

Overall we have found the checks to be somewhat effective in blocking spam and tremendously effective in fostering the notion that we are actively doing something to curb the spam problem.

Notes

- [1] Claus Assmann’s rulesets can be found at `<http://www.sendmail.org/%7Eca/email/check.html>`.
- [2] Robert Harker’s rulesets can be found at `<http://www.harker.com>`.
- [3] Costales, Bryan and Allman, Eric. *Sendmail*. O’Reilly & Associates, 1997, p. 512.

Appendix: The Extended_check_rcpt Ruleset

```
#####
### Extended_check_rcpt - called after check_rcpt. Apply DNS and spam db
### checks on a per-user basis.
#####
SExtended_check_rcpt

# First expand $&f to get the sender's address.
R$* $: $1 $! $>3 $&f
# Apply various rewrites to get workspace into the format:
# sender $! recipient $! username
R$* $! $* $: $2 $! $>3 $1
R$* <@ $* > $! $* $: <$1 @ $2> $! $3
R$* $! $+ < @ $* > $* $: $1 $! $2 < @ $3 > $4 $! $2
# Now look into our username to see what kind of checks the user wants
R$* $! $* $! $* $: $1 $! $2 $! $(spamuser $3 $:<?NOKEY> $)

# No such user in database. Don't do any checks
R$* $! $* $! <?NOKEY> $@ $2

# See if user wants domain checking or both and apply check.
R$* $! $* $! <?BOTH> $: $1 $! $2 $! <?BOTH> $! $>My_check_domain $1
R$* $! $* $! <?DOMAIN> $: $1 $! $2 $! <?DOMAIN> $! $>My_check_domain $1

# Check results of domain check.
R$* $! $* $! <?BOTH> $! $#error $@ 5.1.8 $: "501 Sender domain must exist"
R$* $! $* $! <?DOMAIN> $#error $@ 5.1.8 $: "501 Sender domain must exist"
R$* $! $* $! <?BOTH> $! $#error $@ 4.1.8 $: "451 Sender domain must resolve"
R$* $! $* $! <?DOMAIN> $#error $@ 4.1.8 $: "451 Sender domain must resolve"

# Clear workspace of DNS check results.
R$* $! $* $! $* $! $* $: $1 $! $2 $! $3

# See if user wants database checking or both and apply check.
R$* $! $* $! <?BOTH> $: $1 $! $2 $! <?BOTH> $! $>My_check_db $1
R$* $! $* $! <?DB> $: $1 $! $2 $! <?DB> $! $>My_check_db $1

# Check results. If OK then return original recipient addr and exit.
R$* $! $* $! <?BOTH> $! <OK> $@ $2
R$* $! $* $! <?DB> $! <OK> $@ $2

# User didn't want database check, so return original recipient and exit.
R$* $! $* $! $* $@ $2

# Otherwise, recipient found in spam database, return error from database.
R$* $! $* $! $* $! $* $@ $4

#####
### Supporting Rulesets for Extended_check_rcpt. Mostly stolen from
### stock sendmail package and broken up into separate rulesets.
#####
SMY_check_db
# check for deferred delivery mode
R$* $: < ${deliveryMode} > $1
R< d > $* $@ deferred
R< $* > $* $: $2

R<>$@ <OK>
R$* $: <?> $>Parse0 $>3 $ make domain canonical
R<?> $* < @ $+ . > $* <?> $1 < @ $2 > $3 strip trailing dots
# lookup localpart (user@)
R<$+> $* < @ $+ > $* $: <USER $(spammer $2@ $: ? $) > <$1> $2 < @ $3 > $4
# no match, try full address (user@domain)
R<USER ?> <$+> $* < @ $* $: <USER $(spammer $2@$3$4 $: ? $) > <$1> $2 < @ $3 > $4
# no match, try address (user@domain)
R<USER ?> <$+> $+ < @ $+ $: <USER $(spammer $2@$3 $: ? $) > <$1> $2 < @ $3 > $4
# no match, try (sub)domain (domain)
```



```

R<USER ?> <$+> $* < @ $+ $: $>SpammerLookUpDomain <$3> <$1> <>
# check unqualified user in access database
R<?> $* $: <USER $(spammer $1@ $: ? $) > <?> $1

# retransform for further use
R<USER $+> <$+> $* $: <$1> $3

# check results
R<?> $* $@ <OK>
R<OK> $* $@ <OK>
R<DISCARD> $* $#discard $: discard
R<REJECT> $* $#error $@ 5.7.1 $: "550 Access denied"
R<$+> $* $#error $@ 5.7.1 $: $1 error from access db

SMY_Local_check_domain
SMY_check_domain
R$* $: $1 $! $>"My_Local_check_domain" $1
R$* $! $#$* $#2
R$* $! $* $@ $>"My_Basic_check_domain" $1

SMY_Basic_check_domain
# check for deferred delivery mode
R$* $: < ${deliveryMode} > $1
R< d > $* $@ deferred
R< $* > $* $: $2
R<> $@ <OK>
R$* $: <?> $>Parse0 $>3 $1 make domain canonical
R<?> $* < @ $+ . > $* <?> $1 < @ $2 > $3 strip trailing dots

# handle non-DNS hostnames (*.bitnet, *.decnets, *.uucp, etc)
R<?> $* < $* $=P > $* $: <OK> $1 < @ $2 $3 > $4
R<?> $* < @ $+ > $* $: <?> $(resolve $2 $: $2 <PERM> $) > $1 < @ $2 > $3
R<?> $* <$->> $* < @ $+ > $: <$2> $3 < @ $4 > $5

# handle case of @localhost on address
R<$+> $* < @localhost > $: < ? $&{client_name} > <$1> $2 < @localhost >
R<$+> $* < @localhost.$m $: < ? $&{client_name} > <$1> $2 < @localhost.$m >
R<$+> $* < @localhost.UU $: < ? $&{client_name} > <$1> $2 < @localhost.UUCP >
R<? $=w> <$+> $* <?> <$2> $3
R<? $+> <$+> $* $#error $@ 5.5.4 $: "553 Real domain name required"
R<?> <$+> $* $: <$1> $2

# retransform for further use
R<USER $+> <$+> $* $: <$1> $3

# handle case of no @domain on address
R<?> $* $: < ? $&{client_name} > $1
R<?> $* $@ <OK> ...local unqualified ok
R<? $+> $* $#error $@ 5.5.4 $: "553 Domain ...remote is not
# check results
R<?> $* $@ <OK>
R<OK> $* $@ <OK>
R<TEMP> $* $@ <TEMP>
R<PERM> $* $@ <PERM>
R<RELAY> $* $@ <RELAY>
R<DISCARD> $* $#discard $: discard

SSpammerLookUpDomain
R<$+> <$+> <$*> $: < $(spammer $1 $: ? $) > <$1> <$2> <$3>
R<?> <$+.$+> <$+> <$*> $@ $>LookUpDomain <$2> <$3> <$4>
R<?> <$+> <$+> <$*> $@ <$2> <$3>
R<$*> <$+> <$+> <$*> $@ <$1> <$4>

###
### END of Extended_check_rcpt package
###

```


building a Linux certification program

While great debate goes on within SAGE circles as to whether or not certification for system administrators should occur, the issue within the world of Linux is not if certification will occur, but by whom. At least four separate efforts to establish Linux certification are underway. I will describe one of those programs, which emerged out of a series of mailing-list discussions happening over much of the last year.

The program is called the Linux Professional Institute (LPI). Our mission statement, as stated on our Web site <http://www.lpi.org/>, is:

We believe in the need for a standardized, multi-national, and respected program to certify levels of individual expertise in Linux. This program must be able to satisfy the requirements of Linux professionals, as well as organizations which would employ or contract them.

Our goal is to design and deliver such a program from within the Linux community, using both volunteer and hired resources as necessary. We resolve to undertake a well-considered, open, disciplined development process, leading directly to the establishment of a recognized and widely-endorsed Linux certification body.

I will explain our history, our current program, and where we are going – and invite you to assist us in getting there.

The Past

In early 1998, the Canadian Linux Users' Exchange (CLUE) found that a large number of their users were interested in the idea of Linux certification. Starting in April 1998, they established a mailing list, under the coordination of Evan Leibovitch, focused on certification. They progressed quite far in discussing how a certification program might be implemented. The list grew rapidly and came to include people from around the world. At one point, their list included representatives of three Linux distributions: Caldera, S.u.S.E., and Debian.

Unaware of the CLUE effort, I wrote an article for the October 1998 issue of the *Linux Gazette* (<http://www.linuxgazette.com/issue33/york.html>). In the article, I outlined the reasons I felt a certification program would help the growth of Linux, and I encouraged people to contact me either to point me to programs underway or to help start such an initiative. The response was tremendous, and we immediately began establishing a mailing list to help coordinate our discussions. Along the way, we discovered other individuals and groups who were also working on certification and tried, not always successfully, to find ways to work together on Linux certification.

Last November, Jon "maddog" Hall of Linux International introduced Evan and me to each other. We immediately saw the similarities between our two efforts and explored ways to combine the energies of our two groups to work together on a common program. As we merged our groups and continued to move forward, the initiative attracted a highly talented pool of volunteers, many of whom contributed (and continue to contribute) very long hours to bringing our collective program to reality.

The LPI Certification Program

After long discussions, our program committee, under the leadership of Tom Peters, developed the overall scope of our certification program. We have decided that the



by Dan York

Dan is a member of the board of directors for the Linux Professional Institute. He is also a member of the SAGE certification committee.

[<dyork@lodestar2.com>](mailto:dyork@lodestar2.com)

The Linux certification program must be distribution-neutral and vendor-neutral. The program should not be biased toward any one Linux distribution, nor toward any vendor of education or other services.

The cost of attaining Linux certification should be as low as possible. . . . Whatever mechanism we develop for delivering Linux certification must be global in scale.

program will consist of three levels of certification, although at the time of this writing (April 1999), the names of the levels have not yet been finalized.

At the first level, the candidate will take one exam on basic Linux system administration and a second exam focusing on distribution-specific information. We will create separate exams for each of the major distributions, including Red Hat, Caldera, Debian, S.u.S.E., Slackware, and Pacific HiTech. These distribution-specific exams will address issues such as installation, package management, GUI administration tools, and file locations.

In the second level, the candidate will take two exams. One will focus on advanced system-administration commands, while the other will address Linux internals. All candidates will take the same two exams.

At the third level, we recognize that most system administrators tend to specialize as they gain more knowledge and experience. They tend to become administrators of databases, mail servers, Web servers, or firewalls. For this reason, the candidate will take two exams from among a pool of electives. The final list has not yet been determined, but will no doubt include the topics mentioned earlier.

A complete description of our program is available at [<http://www.lpi.org/program.html>](http://www.lpi.org/program.html).

The Present

Throughout 1999, our committees have been extremely active developing our program and laying the foundation for our future efforts. Overall, our discussions have agreed on the following points:

The Linux certification program must be distribution-neutral and vendor-neutral. The program should not be biased toward any one Linux distribution, nor toward any vendor of education or other services.

The cost of attaining Linux certification should be as low as possible. Costs of exams should be targeted at only what's needed to cover delivery of the exam, with perhaps a slight portion helping to offset development of future exams.

Whatever mechanism we develop for delivering Linux certification must be global in scale. People in any nation must be able to take exams toward certification.

Candidates should be able to prepare for certification through multiple means. Candidates should be able to prepare by reading books, participating in instructor-led classes, using computer/Web-based training, or just working on their own Linux systems. Our certification program should not require any single education source.

The development of the overall certification program (although not necessarily the actual exam questions) should be pursued using open, democratic, and community-based methods.

Today we have a committee structure, based on mailing lists, that is continuing to design and implement our plans on several levels. We are working with computer-based testing vendors to be able to deploy our exams globally. We also have further defined our certification program.

A major part of our time in early 1999 involved a large job-analysis survey conducted across the Web. Scott Murray, the head of our exam-development committee, has experience and education in psychometrics and, working with others in our group, he designed a comprehensive system to conduct a survey of tasks that people do on a daily basis in Linux system administration. After Evan publicized our survey, we had well

over 1,200 people participate in the survey process. The survey, which is just being finished as this article is being written, will guide us in constructing the objectives for the first exams.

Another major component of our recent work has been the construction of an advisory council to provide feedback on the direction of our development efforts. To ensure that our program does meet the needs of the Linux community as well as of organizations that would be hiring Linux-certified people, we asked appropriate individuals and organizations to join our advisory council. While a full list may be found on our Web site, at this time our council includes representatives of major distributions (Caldera, Red Hat, Slackware, and S.u.S.E.), Linux International, the *Linux Journal*, UniForum, the SAGE certification committee, and other information-technology-related organizations.

During this time, we also continued to build our communication with the SAGE certification committee. We have shared information between our efforts and have designated individuals to act as liaisons between the programs. We see similarities between our goals and are eager to cooperate to see if we can build on each other's successes. Unfortunately, with other organizations also working to create a Linux certification program, we cannot afford to wait until the SAGE program can be implemented. Still, it is our hope that as both programs evolve there can be a fit between them.

Finally, we have begun the process of establishing a formal nonprofit corporation and also of seeking financing through a corporate sponsorship program.

The Future

Over the next few months, we will be finishing the development and deployment of much of the first level of our exams. As we complete those efforts, there will be still more distribution-specific exams to implement. We also will begin the development of our second- and third-level exams. Through all this, we will be focusing on marketing the program and finding partners interested in promoting our program and message.

Our challenge is quite different from that faced by the SAGE certification committee in several ways. First, while there are differences among Linux distributions, they are relatively minor compared with the differences among versions of UNIX. Second, there has been very little resistance to the concept of Linux certification within the larger Linux community. Part of that may stem from the high number of Windows converts who have seen what certification has done for Microsoft. Finally, there are market pressures, in that several other entities are developing Linux certification programs.

How You Can Help

We want to make sure this certification program is different from and better than other IT industry certification programs. To that end, we ask for your assistance in helping us build the program. We have well over 200 people on our mailing lists, and there is always room for more assistance. Please visit our Web site (<http://www.lpi.org/>) and join in our discussions. Please also watch our Web site for other opportunities to participate. For instance, in May and June 1999 we will be seeking volunteers to participate in our alpha- and beta-testing of our first exams. Please join us!

Conclusion

Within the world of Linux, certification for individuals will definitely occur. The question is who will do the certifying. We created the Linux Professional Institute and developed our program because we believe that such certification should be given by a nonprofit entity with support from members of the Linux community. We believe that Linux certification should not be something handled by a single vendor or company.

RESOURCES RELATING TO LINUX CERTIFICATION

A more complete and updated list may be found on the "Links" section of the LPI Web site:

<http://www.lpi.org/>

Articles relating to certification:

<http://www.linuxgazette.com/issue33/york.html>

<http://www.linuxgazette.com/issue34/york.html>

<http://www.linuxgazette.com/issue35/york.html>

<http://www.linuxgazette.com/issue37/york.html>

Linux Training Resources:

<http://www.linuxtraining.org/>

Linux Journal Forum:

<http://www.linuxjournal.com/HyperNews/get/certification.html>

Other Linux certification programs:

SAIR, Inc.

<http://www.linuxcertification.org/>

DigitalMetrics

<http://www.digitalmetrics.com/>

Red Hat

<http://www.redhat.com/>

Still, there are other programs underway (see the Resources), and it may ultimately be the market who decides who has the strongest and best certification program.

It's been an exciting experience over the past months that has definitely shown the value and power of a community-based program. The number of talented volunteers who have stepped forward to assist has truly been inspiring. We invite SAGE members who are interested to join with us and help us create a certification program that truly represents the best that we in the Linux and UNIX community can offer.

how-to

Set Up an Apache Web Server

by Adam M. Donahue

Adam is owner and chief technologist of Donahue Consulting, a consulting firm in Manhattan.

<adam@donahueinc.net>

This How-To describes how to install and configure an Apache Web server. It also includes pointers to resources on the myriad options that this Web server offers the system administrator.

As of this writing, Apache is the most popular Web server on the Internet[1]. Part of Apache's appeal is the wealth of add-ons and extensions available for the server. These are referred to as "modules." The core Web server is rather useless by itself. Thus, the basic distribution includes what are considered the "essential" modules: those handling access control and authentication, for example, as well as modules that activate CGI awareness, aliases, server-side includes, and logging. The breadth of modules available means you can usually find the solution to a particular problem, even something highly out of the ordinary. (A searchable database of modules is available at <<http://modules.apache.org>>.) Though even a brief explanation of each module is outside the scope of this How-To, we will take a look at how to activate individual modules after the initial configuration. A need for additional functionality will no doubt arise as new challenges present themselves to you.

Let's start building the basic server. You will need:

- a UNIX machine, built (and properly secured). Apache is available for NT, but we will focus on the UNIX flavors here.
- root privileges if you wish to install the server in a standard location, as well as use the preassigned HTTP port.
- working tools, in your path:
 - = sh
 - = gzip
 - = tar
 - = C compiler
 - = make
- Perl (version 5.004 or greater) is highly recommended; some of the utilities that come with Apache require it. You'll also find it difficult to run many existing CGI-enabled programs without Perl. It has become the de facto language of the Web.

Steps

For most of this installation, you should be the superuser. However, much of this is applicable to a nonprivileged user working from his home directory. I will assume you are installing the server on a machine meant to act as a dedicated Web server, and this requires access to privileged port numbers and nonpublic directories.

1. Obtain server software.

Apache has an excellent track record of reliable releases. You should always download the latest production source release. As of this writing, that is version 1.3.6. A few changes were made to the installation process as of this version, so be sure to use it or – by the time this issue of *login* finds its way into your mailbox – a more recent release.

If you've got Lynx, a quick "source dump" will grab Apache. Run this from a scratch directory:

```
% lynx -source http://www.apache.org/dist/apache_1.3.6.tar.gz \
> apache_1.3.6.tar.gz
```

Note that the Apache Group also provides binary distributions for an array of platforms. Unfortunately, these lag behind the most up-to-date versions by two or three minor release numbers. It's a better idea to download and compile the source yourself, as we'll do here. If you download a binary and later wish to add a new module, you'll need to recompile. So you might as well compile from the get-go. Of course, if you do not have access to a compiler, then the binary version is your only alternative.

Before continuing, become the superuser.

2. Create Web user and group.

It's a good idea to create an underprivileged user and group that you can use to run the server. This has to do with security. Files are read by – and CGI programs executed as – the Web-server process owner. If this owner is root, you leave yourself open to misconfiguration vulnerabilities. An incorrectly written CGI-based program, for example, could allow outsiders to issue commands on your system as superuser! By using an underprivileged user you avoid many of these types of problems.

I use the username `httpd` and the group `web` on my machines, but it isn't important which names you choose. Make sure the user does not have access to any privileged files. In addition, set the Web user's shell to `/bin/false` so no one can log in as that user. (`/bin/false` is a standard UNIX utility that does nothing but return a false value to the caller; any user logging in with it as her shell will immediately be logged back out.) The home-directory setting also is not relevant, though for consistency's sake you may wish to set this to the base directory of the Web-server software.

(How to create a user for the various UNIX platforms is outside the scope of this How-To. I'll assume you've created the necessary user. Also note that Web security is a large issue that involves much more than simply creating a special `httpd` user. I recommend Garfinkel and Spafford[2] and Rubin[3] for more information on Web security.)

3. Decide on location of server source code.

You'll be unpacking the Apache source. This tree contains the code needed for the core server, as well as subdirectories for each major module. Later, when you add new modules, you can simply "attach" them right into this tree, recompile Apache, and reinstall the binary. I find it easier to manage my production server by keeping the source tree completely separate from the installation tree. For example, for many installations I use `/usr/local/src` as the base directory of the source tree, and `/usr/local/web` or

It's a good idea to create an underprivileged user and group that you can use to run the server. . . . If the Web-server process owner is root, you leave yourself open to misconfiguration vulnerabilities.

The base Apache installation is around 2.5MB. You should allocate at least twice that much space on your installation partition. . . . Apache lets you configure it to serve documents from (and write log files to) any directory in your directory tree (as well as other places, like database files).

/usr/local/apache as the installation's base directory. Whatever you decide, move the distribution tar file to the desired source directory before continuing on to the next step:

```
# mv apache_1.3.6.tar.gz /usr/local/src
```

Then change to that directory:

```
# cd /usr/local/src
```

4. Extract Apache source code.

We'll now extract the source from the tar file. At this point you should consider the amount of disk space required for the server source. (This is different from that required for the fully installed server, which is discussed below.) The Apache distribution is now at 1.3MB. The extracted source tree is about 7.5MB in size. You should think about leaving an additional 10MB of space on the source partition dedicated to Apache. This will allow you to add additional modules and options directly into the Apache source tree later on. Apache upgrades occur frequently. In case you want to keep each upgrade in its own directory (e.g., `apache_1.3.3`, `apache_1.3.4`, and so forth), make accommodations for about 20MB of space per distribution. The bottom line: always leave room for growth.

Make sure you're in the root of the source directory (for example, `/usr/local/src`), and execute:

```
# gzip -dc apache_1.3.6.tar.gz | tar xfv -
```

This will create a subdirectory, `apache_1.3.6`, where 1.3.6 is the release number. Now move to that directory and get ready to compile:

```
# cd apache_1.3.6
```

5. Configure compilation options.

Apache comes with a `autoconfig`-style configure program that allows you to activate and/or remove modules and specify other configuration settings. The main thing you need to consider at this stage is where you want your production server to go and how much space this server will require. A production server typically includes both static and dynamic files. The static files include the server binary executable, other binary utilities, and (in general) the configuration files. The dynamic files include your Web content and log files. The base Apache installation is around 2.5MB. You should allocate at least twice that much space on your installation partition. The space required for your Web documents and log files is highly dependent on your particular situation. The good news is that Apache lets you configure it to serve documents from (and write log files to) any directory in your directory tree (as well as other places, like database files). That means that at this point you don't need to worry about whether you have enough space on your installation partition to hold megabytes of content. You can dedicate additional partitions to content later, flexibly and easily, through aliases or other mechanisms. The same goes with log files. The default log-file location can be changed to whatever you deem fit at any time.

As I mentioned, I typically use `/usr/local/apache` (which the Apache Group recommends) as the server installation location. With this base directory in mind, do the following:

```
# ./configure --prefix=/usr/local/apache
```

The `--prefix` is one of the few options you need to concern yourself with at this point. It's used to set the base *installation* directory of the server. Later, when we run an `install`,

all the necessary files for this particular server instance will be copied to that directory. This makes it easy to compile several servers from the same source tree, some for production, some for testing. You need only run `configure` again with a new prefix and rerun the install. Any other existing installations will not be affected.

Running `configure` results in a series of messages explaining the compiler options it's setting, as well as the makefiles it is generating. When `configure` has exited, you are now ready to compile:

```
# make
```

Now install the files needed to run the server into the appropriate installation directory.

```
# make -n install
```

The `-n` switch to `make` lets you see what that invocation of `make` would do without having it actually do it. Take this opportunity to ensure that the install paths check out and that any of the utilities used during the install are referenced in their proper locations.

If everything looks right, do the real installation:

```
# make install
```

After running this, a Web-server instance has been installed in the directory specified in the `-prefix` option above. The following tree is what results (off of the server installation root):

```
conf/      - configuration file
htdocs/    - web pages
cgi-bin/    - CGI-based programs
bin/       - server executable and utilities
logs/      - server logs
icons/     - icons for directory listings
man/       - man pages
include/   - include files
```

(Note that as of 1.3.4, several different directory layouts are possible. You can configure this with the `-with-layout` option to the `configure` program. I will assume you stuck with the default for this How-To. Those who wish to explore other layout options should take a look at the `config.layout` file that comes with the distribution.)

Before continuing, change the ownership of these files as appropriate. You may wish to leave them owned by root, though your Web documents will generally be maintained by you or a group of people, so you might want different ownership settings for them to reflect this. The main thing to remember is that the Web server user must be able to READ these files in order to serve them up on the Web.

The main configuration file for Apache is located in the `conf/` directory and is known as `httpd.conf`. In prior versions of Apache, there were in fact two additional files, `srn.conf` and `access.conf`. Most people found maintaining the three files a bit tedious. Also, which directives belonged in each of the three files was hazy at best. As of 1.3.4 the recommended setup is to use a single file and, as you might expect, this is the default.

`httpd.conf` as provided with the Apache distribution is usually properly configured as part of the build and installation process. If you plan on using port 80 as your server port – which is highly recommended, as that's the port assigned by IANA – then you're practically ready to launch your server at this point. There are a couple of things you will first need to change in `httpd.conf`. But before that you should know a little about how Apache configuration works.

The `-n` switch to `make` lets you see what that invocation of `make` would do without having it actually do it. Take this opportunity to ensure that the install paths check out and that any of the utilities used during the install are referenced in their proper locations.

Open up to the URL

http://hostname/ and you'll find a document included with the distribution that says, "It worked." If you see this message, your server is running successfully.

Apache configuration files are made up of directives. There are two directive styles: single-line directives and container directives. Single-line directives are lines consisting of a directive name followed by one or more arguments. For example, to configure the root directory for your Web pages, you use the `DocumentRoot` directive:

```
DocumentRoot /usr/local/apache/htdocs
```

Other single-line directives that you should be aware of include:

```
ServerRoot - the path to your installation directory base
Listen     - which IP addresses and ports to listen on
User       - the user the web server is running as
Group      - the group the web server is running as
```

`DocumentRoot`, `ServerRoot`, and `Listen` are set up correctly as part of the installation process. The installer, however, does not know about your Web-server user and group. Thus the latter two directives above need to be updated. Move to the configuration directory and edit `httpd.conf` using your favorite text editor:

```
# cd /usr/local/apache/conf
# vi httpd.conf
```

Change the `User` and `Group` directives to read:

```
User httpd
Group web
```

or whatever you decided to call your Web-server user and group. (The order of the directives is not important; notice that the `User` and `Group` directives are already present, however, so you should simply edit their existing values.)

The other type of directive is what I call a "container" directive. It resembles an HTML tag set, with a start tag and a corresponding close tag. The most common use of this type of directive is to configure options on a directory-by-directory basis. A typical entry looks like:

```
<Directory /usr/local/apache/htdocs>
  Options FollowSymLinks ExecCGI
  AddHandler cgi-script .cgi
</Directory>
```

Note how the directive name, in this example `Directory`, is located inside less-than/greater-than signs. It can take options that go within these signs as well. There is also a corresponding close tag for this directive. Inside, we put options that are relevant to the parent directive. For `Directory`, this means options that apply to that directory. Other container directives include if-like statements and virtual-host configurations.

You do not need to edit any of these directives for the basic server installation.

6. Start the server.

Move to the server root directory and type:

```
# bin/apachectl start
```

`apachectl` is an included utility that acts as a front end to the server executable, which is known as `httpd`. It acts similarly to the SVR4 `init.d` scripts. If all went well, you should be able to access your host with a regular Web browser. Open up to the URL `http://hostname/` and you'll find a document included with the distribution that says, "It worked." If you see this message, your server is running successfully.

7. Clean up the document root.

Inside `htdocs/` is a series of pages that ships by default with the Apache distribution. These pages include the welcome message you just viewed and a copy of the Apache manual. You don't need either since these files are directly available from the Apache Group Web site[4]. I usually get rid of them in order to start with a fresh slate:

```
# cd /usr/local/apache/htdocs
# ls apache_pb.gif index.html manual
# rm -rf manual index.html apache_pb.gif
```

Now you can create your own home page:

```
# vi index.html
```

8. Add the server to your machine's start-up configuration.

If you want the server to begin each time you boot up your machine, you need to add the commands necessary to do so to your `rc` files. If you run a BSD-like machine, add the following line to `rc.local`:

```
/usr/local/apache/bin/apachectl start
```

(Add error checking as appropriate.)

If you run an SVR4 machine, create a script, say `httpd`, with the line above and place it in your `init.d` directory. Make sure it is executable. Then create a symbolic link from the `rc` directories corresponding to the run levels at which you wish the server to launch automatically. This is usually run levels two and/or three:

```
# cd /etc/init.d
# vi httpd
# chmod u+x httpd
# cd /etc/rc2.d
# ln -s ../init.d/httpd S99httpd
```

Here is a sample `httpd` script:

```
#!/bin/sh

#
WEBBASE=/local/web/apache
[ -f ${WEBBASE}/bin/httpd ] || exit 0

# See how we were called.
case "$1" in
start)
echo -n "Starting HTTP daemon: "
${WEBBASE}/bin/httpd
;;
stop)
echo -n "Stopping HTTP daemon: "
kill -TERM `cat ${WEBBASE}/logs/httpd.pid`
;;
restart|reload)
kill -USR1 `cat ${WEBBASE}/logs/httpd.pid`
;;
*)
echo "Usage: httpd {start|stop|restart|reload}"
exit 1
esac
exit 0
```


Steps 9, 10, 11 (and, inevitably, 12, 13, etc.) are outside the scope of this How-To. Web development is one topic that does not lack resources. I have included a highly biased listing of some of the texts I found handy when first learning Web technologies.

9. Create your Web pages.

See references [5] and [6].

10. Get comfortable with the server logs.

See reference [6].

11. Activate CGI and begin writing CGI scripts.

See references [6] and [7].

References

- [1] Netcraft Web Server Survey <<http://www.netcraft.co.uk/Survey/>>.
- [2] Garfinkel, S., and Spafford, G. *Web Security and Commerce*. O'Reilly & Associates, 1997.
- [3] Rubin, Aviel D., et al. *Web Security Sourcebook*. John Wiley & Company, 1997.
- [4] The Apache Group <<http://www.apache.org/>>.
- [5] Musciano, Chuck, et al. *HTML: The Definitive Guide*. O'Reilly & Associates, 1998.
- [6] Spainhour, S., and Quercia, V. *Webmaster in a Nutshell: Deluxe Edition*. O'Reilly & Associates, 1997.
- [7] Wall, Larry, et al. *Programming Perl*, 2nd ed. O'Reilly & Associates, 1996.

effective Perl programming

A Perl CPANorama

One of the most obviously useful features of Perl and the Perl community is CPAN, the Comprehensive Perl Archive Network. CPAN is an archive of “all things Perl” that is replicated on FTP and HTTP servers all around the world. CPAN is a common repository for the Perl code base, documentation, and contributed software – and, notably, Perl modules that can be used to extend the capabilities of your Perl installation. The aptly named `CPAN` module greatly facilitates the process of building, testing, and installing Perl modules from CPAN.

In this column I’ll discuss the process of installing modules from CPAN and how to use the `CPAN` module to automate the process.

Finding a CPAN

There are CPAN sites all around the world. Some provide access by FTP, others HTTP, and some both. An easy way to find a CPAN somewhere near you is via the CPAN multiplexer at <http://www.perl.com/CPAN>.

This will take you to some CPAN that the multiplexer thinks is “near” you. Once there, if you like, you can inspect the `MIRRORED.BY` file at the top level of the CPAN directory structure. It contains a list of the current official CPAN mirrors. If the mirror that the multiplexer found for you isn’t ideal, pick a different one from the list of mirror sites.

Finding Modules Manually

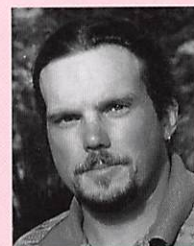
The CPAN is a large and, at first glance (or perhaps always!), somewhat confusing archive. Aimlessly wandering up and down the CPAN directory tree may be entertaining, but such a tactic rarely succeeds as a means of finding a particular module or piece of documentation. The easiest way to begin is usually with the so-called long module list, found in `modules/00modlist.long.html`. The long module list contains a list of all of the modules currently in or formally proposed for CPAN. The list of modules is fairly long, and the information in it is coded for succinctness. Here is an example entry (this one is for my `Sort::Fields` module):

```
Sort::  
::Fields bdpf sort text lines by alpha or numeric fields JNH
```

`Sort::Fields` is the name of the module (broken across two lines to show the hierarchy). `JNH` is the author’s CPAN handle. (You can see a list of handles and the corresponding authors at authors/00whois.html.) `bdpf` is the so-called DSLI code for the module: Development Stage/Support Level/Language Used/Interface Style. In this case, the module is a (b)eta release, supported by the (d)eveloper, written in (p)erl only, with a (f)unction style interface.

Each module is linked into the CPAN several different ways. For example, `Sort::Fields` can be found in all the following places:

```
authors/Joseph_N_Hall/Sort-Fields-0.90.tar.gz  
authors/id/JNH/Sort-Fields-0.90.tar.gz  
modules/by-category/06_Data_Type_Uutilities/Sort/  
Sort-Fields-0.90.tar.gz  
modules/by-module/Sort/Sort-Fields-0.90.tar.gz
```



by Joseph N. Hall

Joseph N. Hall is the author of *Effective Perl Programming* (Addison-Wesley, 1998). He teaches Perl classes, consults, and plays a lot of golf in his spare time.

[<joseph@5sigma.com>](mailto:joseph@5sigma.com)

Because nearly all Perl modules have a standard organization, the process of building, testing, and installing modules is pretty much the same regardless of what the modules do.

Once you have located a module that you want, it's time to install it.

Manually Installing a Module into Your Main Perl Tree

This section assumes that you have permission to modify the contents of your main Perl tree (that is, the place where Perl is installed on your system). If you don't have permission, read this section anyway, then proceed to "Installing a Module into a Nonstandard Place," below.

Because nearly all Perl modules have a standard organization, the process of building, testing, and installing modules is pretty much the same regardless of what the modules do. To install a Perl module, first obtain a copy of the compressed distribution (the .tar.gz file). Unpack the module into a "build" directory (which may be temporary, if you like), for example:

```
# mkdir /tmp/perlbuild
# cd /tmp/perlbuild
# gunzip Sort-Fields-0.90.tar.gz
# tar xvf Sort-Fields-0.90.tar
Sort-Fields-0.90/
Sort-Fields-0.90/Makefile.PL
Sort-Fields-0.90/Changes
Sort-Fields-0.90/test.pl
Sort-Fields-0.90/Fields.pm
Sort-Fields-0.90/README
Sort-Fields-0.90/MANIFEST
```

Move down into the newly created directory, then create a makefile for the module by running the Makefile.PL script:

```
# cd Sort-Fields-0.90
# perl Makefile.PL
Checking if your kit is complete...
Looks good
```

Writing Makefile for Sort::Fields

You have just created a makefile configured to install the module in the "standard" place in your Perl tree. You can now build and test the module:

```
# make
mkdir blib
mkdir blib/lib
(... etc.)
# make test
PERL_DL_NONLAZY=1 /usr/bin/perl -Iblib/arch -Iblib/lib
-I/usr/local/lib/perl5/5.00502/powerpc-machten
-I/usr/local/lib/perl5/5.00502 test.pl
1..38
ok 1
ok 2
(... etc.)
```

If the tests were successful, you now install the module:

```
# make install
Installing /usr/local/lib/perl5/site_perl/5.005/Sort/Fields.pm
Installing /usr/local/lib/perl5/5.00502/man/man3/Sort::Fields.3
(... etc.)
```

That's all there is to it! It's pretty simple, really.

Installing a Module into a Nonstandard Place

If you don't have permission to install a module in your system's Perl tree, or if you

want to install one for your private use only, you have to alter the installation process. The module's makefile defines the location where the module and its supporting files are installed. To change that location, you need to change the makefile. You do this by providing some additional arguments to the `MakeMaker` script that creates the makefile. As an example, here's how I would create a makefile that installs a module into the directories `/home/joseph/perl/lib` and `/home/joseph/perl/etc`.

```
% perl Makefile.PL LIB=/home/joseph/perl/lib \
  PREFIX=/home/joseph/perl/etc \
  INSTALLMAN1DIR=/home/joseph/perl/etc/man/man1 \
  INSTALLMAN3DIR=/home/joseph/perl/etc/man/man3
```

The attribute `LIB` specifies where the module's Perl source code and shared objects (if applicable) will be rooted. The remaining attributes specify locations for the supporting files. (You don't have to specify `INSTALLMAN1DIR` in most cases, but I did so here for the sake of completeness.) After creating an appropriate makefile, build, test, and install the module as normal. The module will be installed in the location that you specified:

```
% make install
Installing /home/joseph/perl/lib/Sort/Fields.pm
Installing /home/joseph/perl/etc/man/man3/Sort::Fields.3
(... etc.)
```

Using a Module in a Nonstandard Place

Because the module in the example above was not installed into its "normal" place in your Perl tree, you have to tell Perl how to find it. The simplest way to do this is with the `PERL5LIB` environment variable. `PERL5LIB` is a colon-separated list of directories that should be searched for Perl libraries. For example:

```
% setenv PERL5LIB /home/joseph/perl/lib
% perl -MSort::Fields -e 'print "loaded successfully!\n"'
loaded successfully!
```

You can also use the `-I` command-line option:

```
% unsetenv PERL5LIB
% perl -I/home/joseph/perl/lib -MSort::Fields \
  -e 'print "loaded successfully!\n"'
loaded successfully!
```

Or the `lib` pragma:

```
% perl -Mlib=home/joseph/perl/lib -MSort::Fields \
  -e 'print "loaded successfully!\n"'
loaded successfully!
```

The `lib` pragma can be used within scripts (although this isn't particularly conducive to portability):

```
#!/usr/bin/perl
use lib '/home/joseph/perl/lib';
use Sort::Fields;
```

Building and Installing with the CPAN Module

The manual procedure for installing modules isn't that difficult, but, like everything else, it becomes tedious if repeated often enough. If things have gotten to that point for you, or if you just generally prefer to avoid busywork, you can use the `CPAN` module to almost completely automate the process of building, testing, and installing modules.

The `CPAN` module is generally used in its "shell" mode. Start it up with the following command:

```
# perl -MCPAN -e shell
```

You can use the `CPAN` module to almost completely automate the process of building, testing, and installing modules.

If this is the first time you have used the CPAN module on this machine, you will get what I like to refer to as “the inquisition,” a series of CPAN configuration questions. If you get the inquisition, answer the questions and stick to the defaults unless you have a reason to do otherwise.

If this is the first time you have used the CPAN module on this machine, you will get what I like to refer to as “the inquisition,” a series of CPAN configuration questions. If you get the inquisition, answer the questions and stick to the defaults unless you have a reason to do otherwise. You will be asked to choose one or more CPAN sites. You should do this with some care – try to put a fast, high-availability CPAN site at the head of your list. In any event, at some point you will enter the CPAN shell and see a message similar to the following:

```
cpan shell - CPAN exploration and modules installation (v1.48)
ReadLine support available (try "install Bundle::CPAN")
cpan>
```

You can type `? or help` at the prompt for a list of CPAN shell commands. One of the most useful is the `i` command, used to obtain information about modules:

```
cpan> i /Fields/
Going to read /usr/local/CPAN/authors/01mailrc.txt.gz
Going to read /usr/local/CPAN/modules/02packages.details.txt.gz
(... etc.)
Distribution  JNH/Sort-Fields-0.90.tar.gz
Module       Sort::Fields  (JNH/Sort-Fields-0.90.tar.gz)
Module       fields       (GSAR/perl5.005_02.tar.gz)
cpan>
```

Building and installing modules with the CPAN module is extremely straightforward. In most cases you can just use the `install` command:

```
cpan> install Sort::Fields
```

This will automatically download, make, and test the module. If it tests successfully, the module will be installed.

You don't have to use the CPAN shell if you don't want. You can use the CPAN module directly from the command line:

```
# perl -MCPAN -e 'install Sort::Fields'
```

This does the same thing as the CPAN shell `install` above.

Installing in Nonstandard Places with the CPAN Module

The CPAN module is perfectly capable of installing modules in nonstandard locations. The mechanism is the same as when you do a manual install – you must change the way module makefiles are built – but to do this with the CPAN module, you have to change the configuration file that the CPAN module uses to issue the command that creates the makefile. There are a couple of ways to go about this, and because of the vagaries of Perl installations and changes in the CPAN module of late, it's impossible for me to say exactly what will be the best procedure for you. However, I'll outline some general approaches.

The CPAN module allows for user-specific private configuration files. These have to be at a specific location – this is hardwired by the module:

```
$HOME/.cpan/CPAN/MyConfig.pm
```

where `$HOME` is the user's home directory. You can create your own `MyConfig.pm` by copying it from your Perl tree:

```
% cp /usr/local/lib/perl5/5.00502/CPAN/Config.pm \
~/ .cpan/CPAN/MyConfig.pm
```

You can make all the changes you need by editing this file, or you can make some later by using the CPAN shell's `o conf` command (see below). However, you will have to

make at least one change before running the CPAN shell. The `cpan_home` parameter in `MyConfig.pm` needs to be changed to point to your home directory. If it isn't changed, the CPAN module will probably not be able to create the lockfile that it depends on, and the module will fail to start up. To make the change, open the file and look for the line defining the `cpan_home` parameter:

```
'cpan_home' => q[/usr/local/build/cpan],
```

Change this to whatever is appropriate, for example:

```
'cpan_home' => q[/home/joseph/.cpan],
```

You also need to make some other parameter changes. The `build_dir` and `keep_source_where` directories should be changed to point to your local `.cpan` directory as well:

```
'build_dir' => q[/home/joseph/.cpan/build],
'keep_source_where' => q[/home/joseph/.cpan/sources],
```

To build makefiles that install in nonstandard locations, you also need to change the `makepl_arg` parameter:

```
'makepl_arg' => q[LIB=/home/joseph/perl/lib \
PREFIX=/home/joseph/perl/etc \
INSTALLMAN1DIR=/home/joseph/perl/etc/man/man1 \
INSTALLMAN3DIR=/home/joseph/perl/etc/man/man3],
```

After you have made these changes, save the file and use the CPAN module as usual. Your modules will now build and install – automatically! – in the locations you have specified.

If you want to change parameter values without hacking on the `MyConfig.pm` file itself, you can use the `o conf` command within the CPAN shell. Use the `o conf commit` command to save your changes:

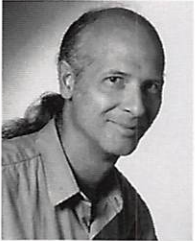
```
cpan> o conf urllist
urllist [q[file:///usr/local/CPAN/]]
cpan> o conf urllist push ftp://ftp.perl.org/pub/CPAN/
cpan> o conf urllist
urllist [q[file:///usr/local/CPAN/], q[ftp://ftp.perl.org/pub/CPAN/]]
cpan> o conf commit
commit: wrote /home/joseph/.cpan/CPAN/MyConfig.pm
```

What's Next?

That's a quick overview of the basics of using the CPAN and the CPAN module. For more information about CPAN, read the various files referred to in the CPAN `index.html`. For more about the CPAN module and the process of making and installing Perl modules, read `perldoc CPAN` and (shudder) `perldoc ExtUtils::MakeMaker`.

In the next column I will show you how to do some neat (and timesaving!) tricks with the CPAN module. Until then, keep reading, coding, and honing your Perl skills!

musings



by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security and System Administrator's Guide to System V*.

<rik@spirit.com>

I am a communist. I admit it. Everyone in my company makes the same amount that I do, regardless of what their position is. The current cultural imperialism – that those who are smarter, or simply more sly, heartless, and aggressive, should make as much as 120 times the pay of lowly workers – strikes me as obscene.

Of course, there is only one person in my company. I just finished last year's books on Saturday, then started vacuuming and emptying the waste cans. So I can afford to pay "everybody" the same thing.

I have visited Eastern Europe and have seen some of the effects of paying everyone the same, regardless of skill level or the effort put into their jobs. I saw a man in Hungary spend an entire day working on a single potted plant in a hotel, empty stores, and frightened people. The leveling effect of Communism did not work very well, eliminating incentive and awarding slackers.

I must admit I enjoyed riding the "free" transit system in Budapest. You were supposed to pay, but nobody did because in the days of Communism, nobody had to. Aspen also has a free public-transit system, although theirs is official.

The idea that free software, or more formally open-source software, is communist, is silly. Under the Communist system, everyone was paid equally, and not very well – unless you were a party official, in which case you received perks. Writers of open-source software do not get paid at all, and there are no party bosses living lives of relative luxury. Most likely these programmers have day jobs, and their bosses live lives of real luxury based on their ability to manage, trick, or coerce people smarter than themselves to work under them.

Illusions

I like working for myself because I am my own boss. Not that I am a very nice boss. I often make myself work strange hours and never pay for overtime. I wonder if I would even get vacations if my wife didn't make me take them.

But I cannot say that I have enjoyed working for other bosses. I have watched several companies go down the tubes, led by bosses driving Porsches (the status car in those days) who hadn't got a clue. Bosses who hired VPs of marketing who told anyone who would listen that the product didn't matter, they could be selling toilet paper. Only marketing matters.

Or bosses who hired very smart engineers, then refused to listen to them. You know the type: You tell them what you think should be done, and six months later, it's their idea, and it gets done. Or you get together with a few co-workers and brainstorm what the most important issues are, only to have the meeting broken up by the bosses. Two years later, those same bosses have realized these issues are crippling the product and demand that they be fixed immediately.

No, I am not Scott Adams. These are real-life experiences from two computer companies where I worked briefly as an employee during the last 20 years. That the current system works well is an illusion, because it could work much better than it does. And open-source software is a model for that process, where groups of programmers work together and new ideas are not put on hold waiting for the boss to get the idea. And there are several teams working on similar projects (GNOME, KDE, etc), and eventually the best will (likely) triumph. The boss in this case is one of the workers.

And note that open-source software can pay very well. Richard Stallman wears ragged clothing to make a political statement. Linus Torvalds, Eric Allman, Paul Vixie (to name a few) are doing very well, as are many other open-source proponents.

More Illusions

And while I am on the topic of money, I'd like you to think about economics for a moment. Not that I am very good at this either, but if the process of concentration of wealth into the hands of a few that began during the Reagan administration continues, we will have a big problem. Trickle-down wealth doesn't work. (Servants don't get paid very well.) The enormous wealth of a few is based on being able to sell product to people, and if the masses have no money, the wealthy will no longer make any money. So the end result of our current economic system might be a subclass that owns everything, or perhaps a collapse.

We could go back to a feudal system, where powerful lords own all the property and "permit" the vassals to work for them. Actually, in some ways we already have, with corporations owning most farms, factories, stores, and businesses. If you are permitted to work for them, you might be able to "buy" a house, so you can spend your income paying interest to another large corporation (the bank).

And what about freedom? Are you free today? What would happen if you decided to study art for the next year or cruise on a sailboat to Panama? Move to a deserted island in the South Pacific or live in Iceland for a year? How about taking off this very afternoon and sitting in a cafe? Could you do any of these things today, or would you risk losing "everything"?

I am not in favor of "overthrowing the system." I watched lots of friends attempt that in the '70s and early '80s, and they obviously did not succeed. But I applaud the notion of exploring new kinds of creativity and alternatives to the way we work. Open source is both of these things, a noble experiment that has already borne fruit. Communism is dead, as dead as those religions that forbid their members sex.

But other economic systems will emerge. It remains to be seen if those systems will be better for most of us or only for a few. My personal belief is that cooperative systems will be the most successful, and that they will also benefit most people, rather than just a lucky (or aggressive) few. And open source represents a model of a cooperative system.

An Accident

In my April Fool column, I stated incorrectly that Linux was ported to more other processor architectures than any other version of a UNIX-like system. Several people wrote email to let me know that I was wrong, and that NetBSD has been ported to many more architectures than Linux.

I am happy to have been corrected and welcome learning by my mistakes. I will not be a boss and ignore my own ignorance, but relish the opportunity to learn. And I hope that some member of the *BSD community will write an article for *login:* that explains as fairly as possible how the four versions of BSD that exist today are different (as well as how they are alike).

Some people pointed out to me that StarOffice already runs under UNIX and permits the reading of MS Word files. It is also likely that if you used that product in cooperation with your email reader, you could be vulnerable to some of the same macro viruses so savored by MS users today. Oh well.

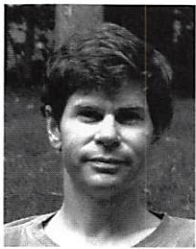
That the current system works well is an illusion, because it could work much better than it does. And open-source software is a model for that process, where groups of programmers work together and new ideas are not put on hold waiting for the boss to get the idea.

The Melissa virus is not the Microsoft Worm I have been predicting. Although it shares some similarities with the original Internet Worm (written by a guy from New Jersey, and attacking homogeneous systems automatically via networks using known security holes), it is not the Big One. We need a slightly larger mass of NT systems for that to occur. It will be amusing, but only from a safe distance.

For now, communist that I am, sitting in a nice hotel room waiting for my stomach to settle before I get in the hot tub, I have just one more thing to say:
WORKERS OF THE WORLD UNITE!

source code UNIX

Security on a Source Code UNIX System



by Bob Gray

Bob Gray is co-founder of Boulder Labs, a software consulting company. Designing architectures for performance has been his focus ever since he built an image processor system on UNIX in the late 1970s. He has a Ph.D. in computer science from the University of Colorado.

<bob@boulderlabs.com>

For more details and some fascinating discussion of government intrusion of privacy see

<<http://www.boulderlabs.com/clipper>>

<<http://www.cs.indiana.edu/classes/a106/readings/clipper.faq.html>>

<<http://csrc.nist.gov/encryption/skipjack-kea.htm>>

The focus this time is security. The primary issues are not unique to computers; society has always had its thieves and vandals. And the solutions are not new – we use a combination of deterrence, monitoring, and penalties for violators.

The appropriate amount to spend on security is a function of the value of your assets and the hostility of your environment. Just as New Yorkers are likely to use bigger, stronger, and more locks than Midwesterners, so too will companies have more elaborate security than the occasional-evening Web surfer.

Most of the measures suggested here are applicable to vendor-supplied, binary operating systems as well as Source Code UNIX. But the latter systems have a big advantage; they can't confuse security with obscurity. Good security is like a well-designed, high-quality lock on a jimmy-resistant door. Sure, it can eventually be overcome by a highly skilled locksmith or by using enough brute force, but we have confidence that those are the only likely methods. Obscurity is like the forgetful bank president who hides the safe combination on the back of his desk blotter.

Protection with obscurity is a disaster waiting to happen. Peter Neumann has reported numerous cases in his "Risks" columns in *Communications of the ACM*. We've seen the famed "Clipper Chip" (encryption with government back-door access) succumb to Matt Blaze's clever black-box techniques at Bell Labs. The reliability of questionable systems can be quickly scrutinized by the community with source code – without it, the scrutiny happens after an incident, with much embarrassment.

Security is hard work; you need to dedicate hours to it every week. Your level of security is a continuum, and everything you do properly will increase the difficulty of penetrating your system defenses. The best situation is one in which one or more people in your organization specialize and can leverage their skills across the company. Consultants can handle your security needs or assist your staff. But even without the experts, there are a handful of things you personally can do to fortify your defenses.

First, alter your mindset; *assume* that your system is not completely secure and then act accordingly. What is it worth to you to be protected against most threats? How badly

will you be hurt by mischief or vandalism? Will your system prevent most of the frequently used attack techniques? What is the value of keeping your system running reliably? Is it a transaction-processing system that must run nonstop? Do you have financials, precious secrets, formulae, inventions, ideas, algorithms, and source code online to protect from competitors? What contingencies do you have in place in the event of a successful denial-of-service attack? For example, the recent Melissa virus didn't directly harm UNIX systems, but some gateways and mail servers became useless as a result of the volume of virus-generated mail.

Careful, deliberate, verified backups provide insurance against disasters. For most of us, a break-in usually wouldn't be a crisis because it is unlikely that the bad guy would know what information was sensitive or would find it. If the bad guy is just curious, you only have to deal with the insult of being violated and the inconvenience of fixing the problem. But there are the angry or revengeful hackers who aim to destroy your information or disrupt your operations.

Recommendations for First-Line Defenses

If high security is required, dedicate specialists to the task. The issues are far too complicated for the "weekend" system/network administrator. Consultants can perform security audits and help fortify your defenses on a continuous basis. If you are on a tight budget and cannot afford to spend much on security, a few simple steps can help prevent the insult of graffiti on your system. I recommend tightening passwords, using secure remote-access tools, clamping down on provided servers, and monitoring for intrusions. Even the home dialup PPP user should implement some of the suggestions below.

Passwords

Establish good passwords for every user. Make sure pseudo users such as uucp, lp, and news have "impossible" passwords (usually designated with * in the encrypted password field). Do you know everyone who has an account on your system? Is that access needed? Firewall or mail-server machines should not allow user accounts. For the remaining accounts on regular systems, educate the users on how to choose a good password. Recently I performed a security audit for a client and uncovered 95% of the passwords in less than a day's worth of PC time. I gave the following advice:

Select a personal, memorable 6- or 8-word phrase, then choose letters from it and permute them. The following phrase is an example:

"5 Sisters, one dog + two hampsters" (yeah, it's true :-)

Taking successively later letters from the words we get

5S,ng+os

If my phrase contained only alphabets, I would permute a couple of letters. For example, you could use "\$" for "S," "@" for "a," etc.

Cracking programs would have required a hell of a long time to guess this password. Software such as Secure Shell (ssh) and Pretty Good Privacy (PGP) allow the user to type long phrases as the cryptographic key. Well-known sayings or movie titles are poor choices for pass phrases. Years ago, Grady Ward wrote in a news article:

"Shocking nonsense" means to make up a short phrase or sentence that is both nonsensical and shocking in the culture of the user, that is, it contains grossly obscene, racist, and impossible or other extreme juxtaposition of ideas. This technique is per-

Assume that your system is not completely secure and then act accordingly. What is it worth to you to be protected against most threats? How badly will you be hurt by mischief or vandalism? Will your system prevent most of the frequently used attack techniques? What is the value of keeping your system running reliably?

I'm against periodically expiring passwords and forcing users to select new ones. I think it encourages people to choose weaker passwords or to write down hard-to-remember ones.

missible because the passphrase, by its nature, ought never to be revealed to anyone with sensibilities to be offended.

If you don't use your password regularly, you might be concerned about forgetting it. (But that's why you picked a memorable phrase as the base.) Don't type your otherwise strong password in a computer file. You might get away with burying the phrase in a text file somewhere, but don't store the permutation in the same place. Also, don't type the password in cleartext over a network, since it's trivial to collect keystrokes on Ethernet and only slightly harder to filter these down into passwords.

All Source Code UNIX systems allow shadow passwords – the encrypted part of the password file is generally not readable. Without the encrypted part, cracking programs are almost worthless because it takes much too long to verify a guess. (A login attempt per guess is required.) As a motivation, recall that the 1988 Morris Internet Worm read password files and hunted for easy pickings from a downloaded list and from the system dictionary file. Don't make it easy.

Create a strong password and be careful with it. I'm against periodically expiring passwords and forcing users to select new ones. I think it encourages people to choose weaker passwords or to write down hard-to-remember ones.

Secure Remote Access

You are likely to use network-connected computers that are under different administrative domains. For example, in addition to accessing machines at Boulder Labs, I connect to the University of Colorado computers, client computers, and computers belonging to relatives. Directory services (such as Novell's NDS or LDAP) and single sign-on systems won't help in these cases. In the past, we comfortably logged on with mechanisms such as rlogin and telnet. Passwords were typed in clear text over the Internet to establish the connection.

Today, these procedures are considered unsafe, because it is easy to intercept passwords along the route. One-time password systems are effective. You've probably seen "security cards" with number sequences that are synchronized with an authentication server. The Security Dynamics cards (SecurID) have six-digit numbers that are valid for about 60 seconds before they are ineffective. Once the user is successfully logged on with a particular code, the code becomes invalid for successive attempts. Therefore, a bad guy who sniffs a code cannot use it. Further, a series of these codes won't help predict the next valid code (at least that is the claim). These types of secure ID cards are somewhat pricey – both for the actual credit-card sized device and for the authentication-server software. Note that one of SecurID's weaknesses is that it is also not "source-code available," which is security by obscurity. If somebody were to crack the sequence-generating algorithm such that sequences could be predicted, the mechanism would be compromised.

S/key, which is freely available, also provides one-time passwords. Your destination host will provide a "challenge" for which you must generate a password on your local computer. That password, which you type over the Internet, is valid only one time. You can precompute passwords for a series of challenges. When I travel, I carry a paper list with me. I establish a partial connection with the remote S/key server, it challenges me, I lookup the corresponding password, and I type it to complete the connection.

One-time passwords are inferior to "encrypted tunnels" because the information you send and receive over the network is cleartext. Virtual private networks or encrypted tunnels hide all of the transmitted data. Essentially all of my remote work is conducted

with Secure Shell (ssh) software. In addition to secure login sessions, you can securely transfer files (scp) or conduct any other client/server transactions, including securely POPing mail and running X11 sessions. (On FreeBSD it's in /usr/ports/security/ssh; it's on the Web at <<http://www.ssh.fi>>.)

Encrypted Mail and Data

To communicate securely with a person over insecure networks, public-key encryption is a good choice. This is the mechanism where a person's *public* key can be published or widely distributed. Once a message is encrypted with the person's public key, it can only be decrypted with the person's corresponding *private* key. (See Greg Rose's PGP Key Signing article in ;login:, <<http://www.usenix.org/publications/login/1998-4/pgp.html>>.) PGP (Pretty Good Privacy) software is a good choice for private communication. Note that PGP also provides "conventional" encryption, in which a *single* key is used both to encrypt and to decrypt a file. This mechanism is a good way to protect files on your computer if you don't have good physical security. It's also a good method for exchanging sensitive data if you have a mechanism for distributing the keys.

Network Services

Inetd is known as the Internet "super-server." It starts at boot time and listens for connections on certain sockets. When a connection is established on one of its sockets, it decides what service the socket corresponds to and invokes a program to service the request. Many vendors ship the configuration file, inetd.conf, with lots of services enabled by default. Because it is hard to be sure that the individual server programs are all perfectly secure, and because inetd itself could have flaws, it is wise to excise as much as possible from this whole mechanism. To improve security, disable any inetd service that your system shouldn't be providing. For example, do you need to provide bootp services? How about POP, NNTP, NTALK, UUCP? If you use ssh and scp, then you may be able to disable telnet, rlogin, rexec, and ftp. The default configuration may enable remote execution of about 40 different programs; consider eliminating most of them. Many people find that they don't need to run inetd at all. (See /etc/inetd.conf and INETD(8).)

A few server processes don't use the inetd mechanism. They start up at boot time and directly listen on ports for service requests. Sendmail is the primary example, but you'll frequently see RPC mechanisms, including NFS, remote print spoolers, system logging (syslog), DNS, and time systems (xntp). Inspect your system to ensure that unnecessary servers don't exist on your system. Look in the bootup scripts (usually /etc/rc.*), use the ps command, and run netstat -a to see what services your system is currently providing over the network. Essentially, if you don't *think* you need it or don't know what it does, then turn it off. Of course, coordinate this carefully with users who may depend on these suspect services. Be methodical when disabling services that you don't understand, so that you do understand the ramifications.

Network Filters and Firewalls

Additional protection is attained by "wrapping" your inetd servers to monitor and filter incoming requests. TCP wrappers provides tiny daemon wrapper programs that log the name of the client host and the requested service. Further, you can specify access control – to restrict what systems can connect to what network daemons. It's available from <[ftp://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.tar.gz](http://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.tar.gz)> or on FreeBSD in the ports collection under security/tcp_wrapper.

Generally, the packet-filtering rules are initiated at boot time. Decisions to accept or reject packets are made early and in the kernel network layer. It's much harder to break into a system if many attacks are thwarted at this level.

TCP wrappers only deals with TCP connections established with the `inetd` mechanism. IP Firewalls (`ipfw`) give you low-level control of all IP traffic into and out of your machine. Generally, the packet-filtering rules are initiated at boot time. Decisions to accept or reject packets are made early and in the kernel network layer. It's much harder to break into a system if many attacks are thwarted at this level. Here are some of the filtering rules from one of our machines:

- 1 allow ip from myIPAddr to any
- 2 allow ip from 123.45.67.89 to any
- 3 allow tcp from any to myIPAddr 25 setup
- 4 allow tcp from any to myIPAddr 80 setup
- 5 allow udp from any 53 to myIPAddr
- 6 allow udp from myIPAddr to any 53
- 7 deny ip from any to any

Rule 1 allows our machine to send IP packets to any other address. Rule 2 gives the 123.45.67.89 machine the right to pass packets through this gateway. Rules 3 and 4 allow mail and http connections. Rules 5 and 6 permit DNS exchange, and rule 7 disallows every other packet. For more details, read the manual pages: `IPFW(8)`.

Firewall and filtering mechanisms don't have to be erected on your general-purpose computers. You can use a commercial router or build your own using software, such as:

- `natd` – Network Address Translation Daemon.
- `fwtk` – FireWall Took Kit, filter services at a level higher than just packet filtering. It provides proxies that can be secured with single-use passwords.
- `PicoBSD` – a standalone, one-floppy-based router with PPP and NATD.

Once your protection is set up, have an external review of your measures. An outside consultant is recommended, but a competent colleague is acceptable. Also, run some of the public-domain security-auditing tools such as `SATAN`, `RSCAN`, and `COPS`. (See the Resource sidebar below.)

Monitoring for Intrusions

After you erect barriers and defenses, monitor for break-in attempts so that you can fortify the weaknesses and even counterattack. Reading system log files, `tcp_wrapper` logs and `ipfw` logs will enlighten you. You'll be surprised who connects or attempts to connect to your machine. If it's questionable, track it down. Make some calls to system administrators at companies or Internet service providers (ISPs) if one of their people is bombarding your machine with attacks. The mail logs and the http logs might also be eye-opening. Report incidents to the Computer Emergency Response Team (CERT, <<http://www.cert.org>>). They also provide tips and advisories for security.

An accomplished hacker is capable of breaking into your system, implanting viruses, back-doors, Trojan horses, or specialized software – and then covering his tracks so that everything appears normal when you examine the log files. How can you deal with this rare but extremely serious threat? You can get a lot of mileage with `tripwire` – it's easy to set up and run. `Tripwire` will traverse specified hierarchies and compute “signatures” for every file. The signatures that you already have available are the file size, permission, owner, and dates. The experienced hacker can replace one of your programs and patch the file size and dates enough to pass a casual filesystem inspection. But as checksums and “message digests” are added to the identity, faking becomes impossible.

Here is how you use `tripwire`. First, on a clean, uninfected system, generate the signatures for crucial hierarchies and save this information in a read-only database or an

offline database. Periodically, run tripwire to recompute the signatures and compare them to the original ones. The differences are flagged. So, if you see that several programs in /bin have new signatures, but nothing about them *should* have changed, you know that something has happened. Once the obvious reasons for their change are eliminated (like another administrator installing new versions), you can suspect the worst and initiate the analytical process of looking for the break-in. Note: here is an application of the 4.4BSD chflags mechanism. You can create a file that is designated as IMMUTABLE – it may not be changed. Similarly, “chflags sappnd” creates a file that may be appended to, but even root processes cannot modify earlier portions of the file. Tripwire is in the ports collection under security/tripwire and available from <ftp://coast.cs.purdue.edu/pub/COAST/Tripwire/tripwire-1.2.tar.Z>.

Purdue’s COAST project, now subsumed by CERIAS, provides for information assurance and security. See <http://www.cerias.purdue.edu>.

Operating System Choice

We see naive users connecting wide-open machines to the Internet. Security mechanisms, which tend to hinder work, need to be set up and administered. The inverse relationship between ease of use and security is likely to continue. Vendors favor shipping easy-to-use, feature-laden systems over secure systems because many end users are not able to handle the additional complexities. Imagine yourself as a sales engineer at MULTIVAQ (SUN, Compaq, Dell, HP . . .) trying to sell a server machine to a small Acme company. At a pre-sales meeting you want to tell Acme that your server is a great value and easy to set up. You *don’t* want to get bogged down discussing the skill required to administer firewalls, eliminate viruses, and patch vulnerabilities. Further, MULTIVAQ doesn’t want its technical service lines jammed with customers that aren’t up to speed on security administration; therefore, the corporate policy tends to favor shipping products with all features enabled, at the expense of security.

OpenBSD boasts that security is one of its principal goals. If a facility cannot be made secure by default, OpenBSD doesn’t incorporate it. The OpenBSD group has a major advantage over many of the other operating-system suppliers. Based in Canada, they are not subject to the ridiculous U.S. crypto export rules. OpenBSD can incorporate strong security mechanisms into their release and ship it anywhere. U.S.-based companies allow only weak cryptography to be exported. Rather than have a U.S. version and an export version, most companies produce the least common denominator. Then *you* must add higher levels of security. For example, you have to separately obtain software such as PGP and 128-bit strong encryption (SSL on Netscape). All of this is in the base for OpenBSD.

Update to Last Year’s “Selecting PC Hardware”

The June 1998 issue of *login:* included my article on selecting PC hardware for Source Code UNIX. (It’s also online at <http://www.usenix.org/publications/login/1998-6/source.html>.) I’ll briefly highlight some of the changes of the last 12 months.

As before, I’ll discuss components in terms of three levels of target systems: LOW, MEDIUM, and HIGH. We’ll define a “system” as a CPU, motherboard, disk, memory, CD, floppy, Ethernet, video/graphics card, keyboard, mouse, power supply, and case. Add a monitor and you have a complete workstation.

The LOW system has dropped in price to around \$600-\$1000, and its speed and capacity have doubled. I like the ASUS P5A Super Socket 7 motherboard with a K6-300 AMD processor. You’ll get a 4-10GB IDE drive, 20-40x CDROM, 4-8MB graphics card,

Resources

<http://www.openbsd.org/security.html>: Security policy and implementation.

<http://www.freebsd.org/security>: Security advisories, tips, guidelines.

<http://www.sans.org>: Tools for security protection, detection, guidelines.

<http://www.cerias.purdue.edu>: Research institute for security.

<http://www.cert.org>: Coordination center for break-ins.

<http://www.eff.org>: Electronic Frontiers Foundation to protect civil liberties, including privacy.

<http://www.daemonnews.org>: On line magazine advocating Source Code UNIX, especially BSD flavors.

<http://www.counterpane.com>: Passwords, crypto essays.

<http://www.crypto.com>: Privacy education site.

Nemeth, Evi; Snyder, Garth; Seebass, Scott; and Hein, Trent. *UNIX System Administration Handbook*. Prentice Hall, 1995.

Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 2nd ed. John Wiley & Sons, 1996.

Once again, Source Code UNIX saves the day. I had an urgent situation that required network booting of a maintenance "miniroot" for a binary-only vendor-supplied system. Both bootp and tftp protocols were needed over the network. I followed the vendor's instructions as best I could but kept getting obscure, meaningless, error messages like "cannot load," "error 7," "failed to boot," "format error." But having Source Code UNIX, I was able to put a couple of printf's into the bootpd and tftpd code and was able to reverse-engineer the process to get the mechanism working.

Thanks to the following reviewers: Tom Poindexter, Mike Durian, Janet Braccio, Barb Dijker, Joel Rem, and Steve Gaede.

32-64MB of S100 RAM and an Ethernet card in the package. In January, we bought a couple of these systems. Their generic memory didn't work under heavy loads. Clocking back to 95MHz seemed to help. Once we switched to high-quality memory, we were able to run again at 100MHz. (We have success with memory from <<http://www.crucial.com>>.) Parity/ECC memory is a bit of a problem with this low system – you have to clock back to 66MHz to get ECC running. For such a cheap system, I'd recommend just getting high-quality memory and running in risky mode.

I've had success with the ultra-inexpensive Spacewalker Shuttle HOT-591P motherboard, AMD K6-2 300, and a Trident 3DImage975 4MB AGP video card. With 4GB disk, 32MB RAM, mouse, keyboard, floppy, CD-ROM, and case it comes to just \$500. I got this system because a local PC shop had all of the items in stock, and they were willing to customize the components for me. From their base system costing \$770, I got credited for Windows 98: (\$70), a Faxmodem (\$30), a 15" monitor (\$130), and a sound card with speakers (\$25).

Many small, local PC shops will work with you to build what you need. I'm sure you can find what you need at the national chains or online shopping, but I have found it hard to get details on what components they are using. The motherboard and video cards are so important but are often not specified. To date, the big guys insist on charging you for Win98, although I expect this will rapidly change.

What was the HIGH system is now the MEDIUM system, and its price range has dropped to \$1200-\$1800. You'll have the fast 100MHz system bus using the 440BX chipset. SCSI disks will give you more performance and reliability. Be sure to get high-quality S100 ECC memory and run with parity or ECC enabled. I like the ASUS P2B series motherboards, which can handle the Pentium II or Celeron processors. The P2B-S has onboard SCSI; you can run 40MB/s Ultra SCSI and/or the newer 80MB/s LVD SCSI (low voltage differential). The P2B-LS has SCSI plus an onboard Intel 10/100 Ethernet.

The HIGH systems have the fastest, most expensive processors and/or multiple processors. You can run with multiple Pentium IIs (or IIIs) or Xeons. There is no need to stay in the x86 line. Linux and BSD systems running on Alphas (DEC/Compaq) are becoming common. There are UltraSPARC ports, and even SGI has publicly announced support for Linux on their hot new Visual PC 320.

General Comments

You save a lot by using processors such as AMD's K6-2/350 or Intel's Celeron 333A instead of the Pentium II. For just a little more you can get the Celeron 433 MHz, which matches the Pentium-II-400 for some benchmarks. (Note that it only runs at the 66MHz bus speed.) SCSI disks continue to be more reliable and higher-performing than IDE disks, but a SCSI controller and the disks will add \$100-\$300 to the cost of the system. Pay attention to the graphics card – while many generic cards will work, you'll spend lots of time fussing with them. At the new 100MHz bus speed, brand-name memory can save you headaches. Try to find a motherboard that will handle ECC at full speed. Monitor prices have dropped 25 percent or more. If you still have a small, fuzzy, flickering screen, treat yourself to a better one with higher quality, bigger size, more resolution, and a faster refresh rate.

the tclsh spot

We spend a lot of our lives extracting information from a mass of data. In our professional lives this is often a case of scanning files or program output for an item of interest. For me, this frequently starts with `grep`, and ends when I've reduced that mass of data to just a few interesting items and perhaps correlated it with some other data.

The student computer lab where I'm teaching this semester is expected to be used only by students who are currently online. However, what with one glitch and another, students frequently end up leaving processes running after they've logged out. Since these tasks chew up resources to no good use, we try to find and kill these tasks.

The algorithm for finding the unwelcome tasks is pretty simple: use `who` to find out who is currently logged into a machine, run `ps` to see what tasks are active, and look for tasks that aren't owned by `root`, `lp`, `daemon`, or the currently logged-in students.

As you might guess, Tcl has a useful set of commands for reading files or program output, manipulating text strings, and reporting results to automate this process. I'll introduce some Tcl commands for I/O and string manipulation, and then show how the application looks.

Tcl I/O commands follow the familiar convention of creating a handle to access the data stream. This handle (called a channel in Tcl) may be used to access a file, device, pipe to another application, or a socket. A channel to a file, device, or pipe is created with the `open` command. For a socket channel, the `socket` command is used. I'll discuss the `socket` command in a future article.

Syntax: `open streamName ?access? ?permissions?`

streamName By default, the name of a file to open. If the first character of the streamName is a pipe symbol "|", then the rest of the name is a program to run attached to a pipe.

?access? The access method: "r" for read, "w" for write, "a" for append. Or a list of POSIX mnemonics including RDONLY WRONLY RDWR APPEND CREAT EXCL NOCTTY NONBLOCK TRUNC The default is "r" (RDONLY).

?permissions? When a file is created, you can declare the permissions mask in numeric form. Tcl supports octal numbers, allowing you to set the modes to values like 0666.

Tcl will substitute a command within square brackets with the result of evaluating that command. Thus, we open a channel to a file with a command like:

```
set inFile [open /etc/passwd "r"]
```

or, to read input from another program:

```
set inFile [open "!who" "r"]
```

Tcl uses the commands `gets`, `read`, and `puts` for I/O. The `gets` command is useful for line-by-line input, while `read` is useful for block reads. The `puts` command will write a single line to a channel.

Syntax: `gets channel ?variableName?`

`gets` reads a line of input from the given channel.

If no *variableName* is present, `read` returns the string of input data.



by Clif Flynt

Clif Flynt has been a professional programmer for almost 20 years, and a Tcl advocate for the past 4. He consults on Tcl/Tk and Internet applications.

<clif@cflynt.com>

The Tcl gets command doesn't generate an error if you try to read past a file's End-Of-File, it just returns a length of -1. Thus, you can read lines from a channel with a loop.

If a *variableName* is present, the input data is placed in the variable with that name, and the number of characters read is returned.

The Tcl gets command doesn't generate an error if you try to read past a file's End-Of-File, it just returns a length of -1. Thus, you can read lines from a channel with a loop like:

```
while {[gets $infd line] >= 0} {  
    # Do Stuff to $line  
}
```

You can also check for EOF with the `eof` command, which returns true when all of the data from a channel has been read. If you use `eof` the read-loop would resemble:

```
while {!eof $infd} {  
    set len [gets $infd line]  
    # Do Stuff to $line  
}
```

Now that data has been read, it's time to process it. The `string` command has several subcommands for manipulating strings, but the "find orphan processes" task uses only a few of these.

Syntax: `string wordend string index`

Returns the index of the character just after the last character in the word that includes the position *index*.

Syntax: `string trim string ?trimChars?`

Trims off all leading and trailing instances of the characters defined by *trimChars*. If *trimChars* isn't defined, then `string trim` trims off whitespace.

Syntax: `string range string start end`

Returns the characters in *string* between the *start* and *end* index markers.

Syntax: `string first string1 string2`

Returns the index of the first occurrence of *string1* in *string2*.

With two of those commands, we can extract the first word from the `who` output (the usernames of the folks currently logged in) with a command like:

```
set name [string range $line 0 [string wordend $line 0]]
```

This name can be added to the list of known names with several commands. One of the easiest is:

```
set nameList "$nameList $name"
```

We can extract the user's login-id from the `ps` output with a line like:

```
set uid [string trim [string range $line 5 14]]
```

which will extract the characters between the 5th and 14th position in the string, and then strip off any spaces.

Finally, we check that this UID is not owned by someone currently logged in with:

```
string first $uid $nameList
```

If the name in `$uid` is not in the string `$nameList`, `string first` will return -1. If `$uid` is in `$nameList` then `string first` will return a value `>= 0`.

Once the data has been read and searched, it's time to format and report the results.

The Tcl `format` is equivalent to `sprintf`. It accepts a `printf`-like format string and a set of arguments, and returns a formatted string.

We can extract the portions of the `ps` output that we are interested in and make a new display with code like:

```
set id [string trim [string range $line 5 14]]
set pid [string trim [string range $line 14 20]]
set cmd [string trim [string range $line 83 end]]
puts [format "%12s - %5d - %s" $id $pid $cmd]
```

The `format` command returns a string that is sent to the standard output device with the `puts` command.

This is the '90s, so we should display our results in a GUI (whether it's appropriate or not).

The simplest way to report a string of results like this is to use the Tk text widget. The text widget is a powerful tool that supports multiple fonts, colors, scrolling, editing, and more. You can insert images and other windows into a text window and can bind actions to events that happen on single characters or large sections of text.

Using the text widget just to display this output is a bit like using a shotgun on a mosquito, but one of the freebies we get with the text widget is the ability to scan up and down the lines of text with `^N` and `^P` as if we were editing in Emacs. This saves me from having to discuss scrollbars in this column.

The text widget has a decent set of defaults, so we could create the widget with a simple command like `text .t`. By default, a text window is 80 characters wide and 24 lines tall. To make life a little more interesting, let's set the size explicitly and use a slightly larger than normal font.

```
set txt [text .t -font {courier 18 bold} -height 10 -width 90]
```

Now, instead of using the `puts` call to display the results, we can use the text widget's `insert` subcommand.

Syntax: `textWidget insert index text`

Insert the *text* at location *index* in the text widget.

The code to run on each machine and find orphaned processes is shown below. In fact, while the guts of the code I use resembles this, I actually run an `expect` script that logs into each machine on the local network and looks for orphaned processes. It reports the data as simple text strings. However, `expect` is a topic for another column.

```
# Open a text window for display.
set outputWindow [text .t -height 10 -width 90 \
    -font {courier 18 bold} -disable ]
pack $outputWindow

# Initialize the namelist with the names of users
# that we know will be online (root, daemon, lp),
# and add "UID" to cleverly remove the header
# from consideration.
set namelist "root daemon lp UID"

# Run who and read the input from the who command.
set infl [open "|who" ]

# The gets call will return -1 when it hits an EOF.
# Read the lines, extract the user name,
# and if the username isn't already in our list, add it.
while {[gets $infl line] >= 0} {
```

One of the freebies we get with the text widget is the ability to scan up and down the lines of text with `^N` and `^P` as if we were editing in Emacs.


```

    set name [string range $line 0 [string wordend $line 0]]
    if {[string first $name $namelist] < 0} {
        set namelist "$name $namelist"
    }
}

# close the file, we're done with it.
catch {close $infl}

# Invoke ps, and read the input.
set infl [open "|ps -elf"]

# This time, we'll use the eof command to check
# for end of file.
while {[eof $infl]} {
    set len [gets $infl line]

    # extract the user id from the line of data.
    set id [string trim [string range $line 5 14]]

    # If the id is not in our namelist, we have a hit.
    # Get the pertinent data and update the window.
    if {[string first $id $namelist] < 0} {
        set pid [string trim [string range $line 14 20]]
        set cmd [string trim [string range $line 83 end]]
        $outputWindow insert end \
            [format "%12s - %5d - %s\n" $id $pid $cmd]
    }
}

```


using java

Remote Method Invocation

An aim of distributed systems is successful interaction among programs running in different address spaces. An earlier article in this series (*;login*, October 1998) discussed RMI, Sun's way of allowing programs written entirely in Java to share information across address boundaries. RMI permits a Java object in one address space to invoke methods contained in a Java object that runs in a separate address space. This can happen in applications in which each object is a thread that is run in its own address space. Another way to run in separate address spaces is to run each Java program on a separate machine. An important feature of RMI is that a method invocation on a local object has the same syntax as that on a remote object.

This article presents a simple RMI example that walks the reader through the steps required to write such applications.

Summary of Java RMI

Remote invocation is nothing new. For instance, C programmers have used Remote Procedure Call (RPC) semantics to execute a C function on a remote host. What makes RMI different is that in Java it is necessary to package both data and methods and ship both across the network. (RPC works on data structures primarily.) This implies that the recipient must be able to interpret the object after receiving it.

RMI at a glance:

The Good

1. It is very easy to use.
2. Remoteable interfaces have a special exception.
3. It supports object-by-value.
4. Versioning is built into serialization.

The Bad

1. Java call semantics are changed so that thread identity is not maintained.
2. Callbacks are blocked in synchronized methods.
3. It is not always intuitive.
4. It is not available for use with other languages.

The Ugly

1. There are limited development tools.
2. Clients need access to latest stubs.
3. Performance can be slow as you scale your application.

Although RMI does not directly support other languages, it is possible to use the Java Native Interface (JNI) to create Java wrappers that can be used with RMI. Of course this introduces yet another level of indirection and may further exacerbate performance problems that are due to scaling.



by Prithvi Rao

Prithvi Rao is the founder of Kiwilabs, which specializes in software engineering methodology and Java training. He has worked on the development of the MACH OS and a real-time version of MACH, and he holds two patents resulting from his work on mobile robots.

<prithvi+@kiwilabs.com>

I have avoided the use of "client and server" because there is really no client-server relationship here. It really is a case of one JVM making another JVM do something. The client-server abstraction exists at the level of the application and not at the level of RMI.

Java RMI Example

The following example demonstrates the use of RMI within applets, which is a typical use of Java and RMI. It is conceivable that an applet (which is a Java program running within the context of a browser such as Netscape, appletviewer, or HotJava) needs to invoke methods on objects that are on other machines. Consider a database application, for instance, with a GUI that is an applet and a data server that is multithreaded and written in Java. The GUI runs on a thin client, and the data server may run on another machine across a network. This is a fairly common scenario and so our example is not atypical.

There are really two Java Virtual Machines (JVMs) involved in this example. The first JVM is the one into which the applet is loaded. The other JVM is the one on which the remote object exists. Let's think of the first JVM (the one running the applet) as the local side. I have avoided the use of "client and server" because there is really no client-server relationship here. It really is a case of one JVM making another JVM do something. The client-server abstraction exists at the level of the application and not at the level of RMI. Another way to look at this is that one JVM is invoking methods on an object running on another JVM.

Let's look at the code for an applet. This is the local side. This file is called `HelloApplet.java`.

```
package example.hello;
```

This means that the `Hello` class is in a package `examples.hello`. Remember that this is interpreted as a directory relative to `CLASSPATH`.

```
import java.awt.*;
```

This is necessary because we are writing an applet, and applets are part of the Abstract Windowing Toolkit (AWT). (See my *login*: April 1999 article for an AWT example.)

```
import java.rmi.*;
```

This is new. `java.rmi` is a package that provides support for RMI, and so we must import it.

```
public class HelloApplet extends java.applet.Applet {
```

We are extending `applet`, except that because we did not import `java.applet` we must provide the fully qualified name (fqcn) for the applet class we wish to extend.

```
String message = "";
```

Field for the message that will be received from the remote object.

```
public void init() {
```

The `init` method for the applet.

```
try {
    Hello obj = (Hello)
        Naming.Lookup("//" + getCodeBase().getHost() +
            "/HelloServer");
```

This creates an instance of the remote object inside the remote JVM. We will say more about this later. It is important to note that this method returns an instance of a class `Hello`. The argument it takes is a URL, or so it seems because of the `//`.

```
message = obj.sayHello();
```

This statement invokes the method `sayHello()` of the remote object.

```
} catch (Exception e) {
```



```
System.out.println("HelloApplet: an exception occurred:");
e.printStackTrace();
```

Naming.lookup throws three exceptions, so we must catch them.

```
}
public void paint(Graphics g) {
    g.drawString(message, 25, 50);
}
}
```

Draw the string that was received from the remote object on the screen.

Now let's take a look at the remote side.

There are two files associated with the remote side. The first is Hello.java.

```
package examples.hello;
```

Put this in the same package as the applet.

```
public interface Hello extends java.rmi.Remote {
```

Define an interface called Hello that extends java.rmi.Remote. We will explain this later. Basically, if an object wants to be a remote object – that is, it wants to be able to be invoked by some other object – then it must implement the interface java.rmi.Remote.

```
{
    String sayHello throws java.rmi.RemoteException;
}
}
```

It must also specify those methods that can be invoked remotely. In this case there is only one such method, sayHello. Recall that the applet HelloApplet calls a method by this name. This is the method that will be invoked. At this point it is merely a method of an interface and has only a signature but no body.

Now the second of the two files for the remote side. This is called HelloImpl.java.

```
import examples.hello.*;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
```

All servers must be subclasses of this if they want to be remote.

```
public class HelloImpl extends UnicastRemoteObject implements Hello
{
```

This implements the interface Hello, and since Hello extends the Remote interface, this makes HelloImpl a class of type Remote.

```
    private String name;
```

This is the name by which this object is known to the other objects that wish to invoke it.

```
    public HelloImpl(String s) throws java.rmi.RemoteException {
        super();
        name = s;
    }
```

The constructor for this class.

```
    public String sayHello() throws RemoteException {
        return("Hello World");
    }
```

Basically, if an object wants to be a remote object – that is, it wants to be able to be invoked by some other object – then it must implement the interface java.rmi.Remote. . . . It must also specify those methods that can be invoked remotely.

The object that invokes methods on this JVM should not be allowed to roam free on the host machine on which this JVM is executing. The RMI security manager enforces a suitable security policy.

Recall that the `sayHello` method is part of the interface `Hello`. Since this class implements that interface, it must provide a definition for it. The definition simply returns the string "Hello World."

```
public static void main(String args[])
```

This application's main function. Recall that applets do not need `main`.

```
System.setSecurityManager(new RMISecurityManager());
```

Remember that applications can set their security manager (;*login*;; August 1998). The object that invokes methods on this JVM should not be allowed to roam free on the host machine on which this JVM is executing. The RMI security manager enforces a suitable security policy. Recall that the applet invoking a method on this object might send it data that might be bytecode that is capable of being executed. For this reason it is necessary to ensure the presence of the security manager.

```
try {
    HelloImpl obj = new HelloImpl("HelloServer");
```

Instantiate a `HelloImpl` and call it `HelloServer`. Recall that the applet used `Naming.lookup()`.

```
Naming.rebind("HelloServer", obj);
System.out.println("HelloImpl created and bound in " +
    the registry to name HelloServer");
```

Register this object as existing and print out some diagnostics.

```
    } catch (Exception e) {
        System.out.println("HelloImpl.main: exception occurred:");
        e.printStackTrace();
    }
}
```

Running the Example

1. Compile the applet and create an HTML page for it (;*login*;; April 1999). The applet runs on the local machine.

```
> javac HelloApplet.java
```

2. Compile the Java classes on the "remote" machine.

```
> javac Hello.java
```

Make sure that this file is in a directory `examples/hello` relative to where the file `HelloImpl.java` exists.

```
> javac HelloImpl.java
```

3. Generate the stubs and skeletons on the remote machine using the `rmi` compiler (`rmic`).

```
> rmic HelloImpl
```

This creates two files called `HelloImpl_Skel.class` and `HelloImpl_stub.class`.

4. Start the RMI registry on the server.

```
> rmiregistry
```

The registry is used to let the two Java objects locate each other and therefore establish contact. Notice that nowhere in any of this code is any direct reference to low-level networking interfaces such as sockets. This is transparent to the user. The registry runs on the remote machine.

The important thing to remember is that the stub is sent from the remote object to the object that invoked it when the local object uses the lookup method of the class

Naming.

5. Start the applet.

Start a browser and load the HTML page for this applet.

Conclusion

The use of distributed objects is fairly common in many IT application domains. Two examples are the health-care industry and the stock market. In most cases where distributed objects are used, it is necessary to create an infrastructure in which Java and non-Java objects can invoke methods on each other. In these cases Java RMI cannot be used without first writing some kind of Java “wrapper” for the non-Java code. These implementations therefore use CORBA or DCOM. If a pure Java application is envisaged, then Java RMI is a good choice for its ease of use and its ability to facilitate the rapid prototyping of the application.

In future articles we will demonstrate the capability of Java with other middleware packages. There is no substitute for being well informed in order to make intelligent decisions, and Java RMI is only one piece of the puzzle.

the bookworm

Books reviewed in this column:

Uyless Black

ATM Volume I

2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.
ISBN 0-13-083218-9

Uyless Black

ATM Volume II

Upper Saddle River, NJ: Prentice Hall, 1998.
ISBN 0-13-571837-6

Uyless Black

ATM Volume III

Upper Saddle River, NJ: Prentice Hall, 1998.
ISBN 0-13-784182-5

Natalie Giroux & Sudhakar Ganti

Quality of Service in ATM Networks

Upper Saddle River, NJ: Prentice Hall, 1999.
ISBN 0-13-095387-3

Rainer Haendel, Manfred N. Huber, and Stefan Schroeder

ATM Networks

Reading, MA: Addison-Wesley, 1994.
ISBN 0-201-42274-3

Berry Kercheval

TCP/IP over ATM

Upper Saddle River, NJ: Prentice Hall, 1998.
ISBN 0-13-768599-8

Kevin Downes et al.

Internetworking Technologies Handbook

2nd ed. Indianapolis, IN: Cisco Press, 1998.
ISBN 1-57870-102-3

Mark Sportack

IP Routing Fundamentals

Indianapolis, IN: Cisco Press, 1999.
ISBN 1-57870-071-X

Mark Tripod

Cisco Router Configuration and Troubleshooting

Indianapolis, IN: New Riders, 1999.
ISBN 0-7357-0024-9

Priscilla Oppenheimer

Top-Down Network Design

Indianapolis, IN: Cisco Press, 1998.
ISBN 1-57870-069-8



by Peter H. Salus

Peter H. Salus is a member of ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past lustrum. He owns neither a dog nor a cat.

<peter@pedant.com>

Well, as advertised, this month it's all ATM, switching, and routing. If this isn't interesting to you, come back for the August issue.

ATM

ATM = asynchronous transfer mode, not the hole in a wall that trades currency for a plastic card. Nor "A Terrible Mistake," either. If you are allergic to 3- and 4-letter abbreviations, read no further.

In 1991 I got a book on ATM by Haendel and Huber; 1994 brought me a revised and augmented edition; 1995 brought me a book on ATM by Uyless Black. The past few months brought me a volume by Kercheval, three by Black (one of them the second edition of the 1995 volume), and one by Giroux and Ganti. So here's an attempt at an overview.

First of all, it's important to know that ATM was the creation of the telephone companies, intended to fit neatly into the ITU's broadband vision of the future. But the data-communication community took a look at that version and decided that they could do better.

ITU-T constructed a model not unlike the famous ISO seven-layer cake. In this model, the ATM layer is immediately above the "physical layer." ATM is *asynchronous* because each station in the net can send or receive as many or as few cells as it wishes. (Cells are the fixed-size packets used in ATM. Thanks to the CCITT they comprise 53 bytes: 5 bytes for the header and 48 for the "payload.")

When ATM came out, I hated the con-

cept. After all, IP is connectionless; if a link goes down, it gets routed around. In ATM, if a link goes down, you're down. But further work has convinced me that ATM will be with us for a long time in some form.

Black's 1995 volume was subtitled *Foundation for Broadband Networks*. The second edition is just a bit larger, waxing from 426 to 446 pages. Volume II is *Signaling in Broadband Networks*; volume III is *Internetworking with ATM*. They are all interesting books, though I thought volume III weaker than the other two.

Structurally, they all carry introductory material, lists of abbreviations, and an excellent list of references. In between they are quite different from one another.

Foundation takes us through an introduction to modern telecommunications to layered communications involving SONET and ATM. We then ease our way by means of ISDN-B into ATM, covering the AAL, switching operations, traffic management, connection control, and internetworking, before returning to SONET. Black then covers network management, the physical layer, and (finally) the business model. He admits that the current ATM market is soft, which few other authors are willing to actually state.

With Black's *Foundations* volume read, you can turn to Vol. II, *Signaling in Broadband Networks*. Here Black details the evolution of the differences between signaling and transport network, the blurring of distinctions between them, and SS7. SS7 (Signaling System 7) is both powerful and flexible, though it does (like POTS) use an out-of-band system for signaling. While it's true that SS7 was intended for physical circuits, it can be modified for employment with virtual circuits too.

Black goes on to describe ISDN-B and then moves at full speed into broadband technology. While he does a fine job of explaining a vast number of topics, he

employs an even vaster number of abbreviations (largely three- and four-letter ones) in his presentation. The result (for me) was that I was continually flipping to the list of abbreviations to ascertain what was being discussed. Sentences like: "ATM uses small, fixed-length units called cells that are identified with VPIs and VCIs that are contained in the cell header" do not increase readability. Nor do chapter titles like "SAAL, SSCOP, and SSCF" (chapter 6). At some point, we've got to chew our way out of the alphabetic spaghetti.

But despite this, I think that Black does an admirable job of presenting the mechanics of broadband signaling as well as presenting a very useful list of possible error messages.

I had more problems with volume III, *Internetworking with ATM*. This was largely because of Black's idiosyncratic use of "internetworking": "Internetworking is the sharing of computer resources by connecting the computers through a number of data communications networks." I prefer Comer's version: "Internetworking accommodates multiple, diverse underlying hardware technologies by providing a way to interconnect heterogeneous networks and a set of communication conventions" (*Internetworking with TCP/IP*, vol. 1, 3rd ed., 1995).

Another problem I had with volume III arose as a result of receiving Giroux and Ganti, an excellent book on QoS under ATM. As I happen to believe that QoS may be the most important question in networking in general, I was much taken by Giroux and Ganti's presentation of ATM traffic management.

As a comparison, Giroux and Ganti devote nearly four times more space to frame relay on ATM networks than does Black. Their exposition is also significantly less riddled by abbreviations and is quite clear and well written. In fact, Haendel, Huber, and Schroeder devote

more space to frame relay than does Black.

These books are very different from one another: they serve more or less the same audience, but in different ways. For example, those of us devoted to TCP/IP will find Kercheval indispensable. Those wanting an overview will want Haendel, Huber, and Schroeder, and Giroux and Ganti. And those wanting a detailed run-down will go for Black's volumes.

Networking

It's interesting to go from Black or Kercheval to the *Internetworking Technologies Handbook*, with its single brief chapter on ATM switching (pages 269-300). While everything is correct here, I had the feeling that I was reading an entry in a biographical dictionary that read something like "Shakespeare, William (1564-1616). English dramatist and poet." But the *Handbook's* nearly hundred pages of glossary are very useful indeed. Unfortunately, fiberoptic cable and wave-division multiplexing aren't among the "internetworking technologies" included here.

In fact, even a few pages on WDM would have been welcome in most of the volumes in this column on ATM and other aspects of networking. Reading the stuff in <http://www.atmdigest.com/WDM.htm> and in <http://www.oiforum.com/index.html> (Optical Internetworking Forum) makes you realize that this is the true technology of the future.

Mark Sportack has written a really interesting "comprehensive introduction to routing concepts and protocols." I enjoyed it a good deal; and I think I learned a lot, too. Sportack calls routing "the most complicated function of a network," and I'd agree with him. But there's a gap in his presentation: not a word on queueing or queueing theory. There should be. Len Kleinrock's two volumes came out over 20 years ago; more recently, there's one by Gross and Harris (3rd ed., 1997) and a short volume by

Kleinrock and Gail (1996).

Tripod has done a really fine job on Cisco routers. I especially enjoyed his chapters on switched circuits, the Cisco router inventory, and troubleshooting. It's a well-written, useful book.

Oppenheimer's volume is a truly worthwhile one. From "Identifying Your Customer's Needs and Goals," through "Logical Network Design" and "Physical Network Design," to the chapters on testing and optimizing, I found myself admiring her presentation and thoroughness.

And that's more than enough words on ATM and networking for a month or two.

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

Free subscription to ;login:;

the Association's magazine, published eight to ten times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

Access to ;login: online

from October 1997 to last month.

<www.usenix.org/publications/login/login.html>

Access to papers

from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web.

<www.usenix.org/publications/library/index.html>

The right to vote

on matters affecting the Association, its bylaws, election of its directors and officers.

Optional membership

in SAGE, the System Administrators Guild.

Discounts on registration fees

for all USENIX conferences.

Discounts

on the purchase of proceedings and CD-ROMS from USENIX conferences.

Savings

(see <<http://usenix.org/membership/membership.html>> for details)

10% off all Academic Press Professional books

10% off BSDI, Inc. "personal" products.

10% off Morgan Kaufmann books.

20% off New Riders/Cisco Press/MTP books.

10% off OnWord Press publications.

10% off The Open Group publications.

20% off O'Reilly & Associates publications.

\$10.00 off Prime Time Freeware publications and software.

10% off Wiley Computer Publishing books.

Special subscription rates

(see <<http://usenix.org/membership/membership.html>> for details)

\$45 subscription to *IEEE Concurrency* (regularly \$88).

15% off subscription to *The Linux Journal*.

\$5 off subscription to *The Perl Journal*.

20% off subscription to any Sage Science Press journals.

For more information regarding membership or benefits please contact

<office@usenix.org>

Phone: 510 528 8649

USENIX news

USENIX Teams Up To Put on the 1999 Atlanta Linux Showcase (Larger USENIX Role Planned for Y2K)

by Cynthia Deno

USENIX Marketing Director

<cynthia@usenix.org>

The Atlanta Linux Enthusiasts, USENIX, and Linux International are pleased to announce co-sponsorship of the 3rd Annual Atlanta Linux Showcase. The Atlanta Linux Showcase will be held at the Cobb Galleria in Atlanta, Georgia. USENIX will offer a Tutorial Program on October 12-13, which will be followed by General Conference Sessions and an Exhibition on October 14-16. The Exhibition is expected to feature 125 Linux product and service vendors, including most of the leading lights.

Next year, USENIX will take the lead in sponsoring the 2000 Atlanta Linux Showcase. Atlanta Linux Enthusiasts and Linux International will continue to support what will become a full-fledged USENIX conference.

The Call for Papers for the 1999 Atlanta Linux Showcase

We invite you to submit proposals (submission deadline: July 1, 1999) to enhance the invited talks, tutorials, and Birds-of-a-Feather sessions. ALS is a forum that brings together both experts and peers in our field. The conference will feature three tracks over three days with 40 speakers discussing both technical and business issues concerning the Linux Operating System.

Details of the Call for Papers are found at the back of this issue of ;login: and online at <www.linuxshowcase.org>.

Incident Cost Analysis and Modeling Project II (I-CAMP II)

by Gale Berkowitz

USENIX Deputy Executive Director

<gale@usenix.org>

Colleges and universities are becoming increasingly concerned about security incidents in the distributed and diverse electronic networks and services they have created on their campuses. This concern is being heard from data handlers, data stewards, data administrators, and system administrators. Decision-makers are often reluctant to invest the required level of resources in security-related functions, simply because they lack information about data security and the costs and benefits associated with it.

Some people at the University of Michigan are concerned enough about security incidents to study them systematically. The USENIX Association is providing funding to conduct the study.

The Incident Cost Analysis and Modeling Project II (I-CAMP II) project is under the direction of Dr. Virginia Rezmierski, Director of the Office of Policy Development at the University of Michigan. It is designed to learn more about the types of information-technology (IT) incidents occurring, how often they occur, and the costs associated with rectifying them. Examples of common IT-related incidents include unauthorized access to data, denial of service, power interruptions, hardware failures, and backup failures.

During the first phase of the project, I-CAMP-I, researchers gathered a sample of IT incidents, developed a cost-analysis model, and reviewed existing IT risk management models in higher education.

The I-CAMP-I report found that:

- IT incidents were occurring at a steady and perhaps alarming rate.
- Managing such IT incidents takes valuable technologist time away from needed production and development responsibilities.
- Real, and in some cases significant, costs are associated with these IT incidents, even when a conservative approach to cost analysis is taken.
- Whereas hacker-type incidents were most readily identified by initial campus contacts, a greater variety of incidents, including data theft, were identified as the project progressed and others became involved.
- Frequency data, not collected in this project, is needed to estimate overall risks and costs to campuses.
- Recommendations for management were needed to begin to reduce or eliminate these costs.

I-CAMP-II expands on the design and implications from the first phase in the following ways:

- It expands the sample of incidents to other representative campuses from among the Committee for Institutional Cooperation (CIC) Big Ten campuses, as well as a select sample of large universities that have incident databases.
- It expands the range of incidents tracked and factors affecting them.
- It develops specific recommendations to reduce or eliminate the risks of identified types of IT incidents.

At the end of the first quarter of the project, I-CAMP-II is well underway. An advisory board has been selected, participating schools have been chosen, and the project methodology has been refined. The project team has already been working on better ways to classify incident types. It has divided existing incidents

into two groups – those determined to be malicious behaviors and those that are seen as unthinking acts. Both categories of incidents jeopardize operations and security and/or add liability to the institutions.

The I-CAMP-II study will provide system administrators with critical information to increase security awareness within their organizations. A final report from the project is expected early in 2000.

For more information about the project, please contact Virginia Rezmierski, Director of the Office of Policy Development at the University of Michigan, at 734.647.4274, or by email at <ver@umich.edu>.

20 Years Ago in USENIX

by Peter H. Salus

USENIX Historian

<peter@pedant.com>

June 1979. Toronto, Ontario.

It's really tough trying to write this up.

For example, Brian Kernighan wrote me:

My memory is quite dim. According to my calendar, I gave a talk at the software tools group in Toronto, and it says that I gave a "v7 talk" at "UNIX user's group," but I have no memory of the latter and hardly any of the former.

David Tilbrook wrote:

Toronto firsts: Rob Pike and David Tilbrook give maiden USENIX speeches together on QED.

And Rob wrote:

That was a very long time ago. I don't remember anything much, although it might come back to me with prodding. I do recall the discussion afterwards included comments from the audience about ex/vi, and there was an exchange about the high cost of screen editors,

which interrupted the operating system for every keystroke.

Thanks to Geoff Collyer (who must throw out even less than I do), I have the program(me)s for STUG and for USENIX, as well as his notes.

STUG, the Software Tools Users Group, met on Tuesday, June 19. Kernighan made some introductory remarks (including the news that UNIX/RT "is not likely to be released for five years"). He was followed by Dave Hanson (then of the University of Arizona) on portable file and I/O systems and Doug Comer (then as now at Purdue) on his "Mouse4," a rewritten preprocessor that used a hash table.

After a break, several of the Georgia Tech group gave a paper on their tools running on a PRIME. (My view of their work is in *A Quarter Century of UNIX*, chapter 12.) Dennis Hall then delivered a status report on LBL's Virtual Shell. He said:

My strongest memory is that we weren't really legitimate. I felt like a pretender. Actually, I never quite got over that feeling. The true blue UNIX people thought we were wasting our time on second best. The true blue VMS (or any Brand X) users thought we were debasing their systems. Remember the person who thought it would be a good idea to build a TSO shell interface for UNIX using the tools?

After lunch, Robert Munn (University of Maryland) gave a status report on his rat-for preprocessor, which was being used to distribute a crystallography course to 250 students.

The afternoon concluded with a "Haves and Wants" session and a discussion of distribution and standards led by Kernighan.

The USENIX conference proper began on Wednesday with a session on languages (Pascal, Euclid, and YASL). After lunch,

there was a session on UNIX, including papers by Brian Kernighan (V7) and Tom London (V32).

Thursday morning began with a talk by Al Arms of Western Electric, who announced the horrendous new fee structure: \$20K for V6, \$30K for PWB, \$28K for V7, and \$40K for V32 – per CPU! He also said that it was forbidden to teach the internals of V7. Not a good showing from the corporate set.

There was a discussion of the user group and of the various compilers available from Whitesmiths’.

After lunch it was text processing and graphics (Dennis Murmaugh, Martin Tuori, Tom Duff, and Bill Reeves). Tom showed a videotape of computer-generated images (512 x 512 x n; where n=8 or n=24). Geoff Collyer recalls that “it was better resolution than TV.”

The afternoon session concluded (as Tilbrook recalled): “Bill Buxton gives concert . . . probably first musical presentations at a USENIX or any other computer conference.” Collyer wrote that it was “very boring.”

In the evening there was a reversi (=Othello) tourney.

Friday began with a session on mini UNIX, MRS database, realtime UNIX (with papers by Neil Groundwater and by Eric Ostrum), networking, and several applications: an RT-11 emulator, an

accounting system, STUG tools, and Idris.

The afternoon session was on system improvements: multiple-address space, supporting 64 terminals and 1,000 users on an 11/70 (Ian Johnstone), large UNIX systems (George Goble), high-performance UNIX (Mike Muuss), and porting UNIX to a Univac (Hale Pierson). Saturday morning was sparsely attended. (Most folks had been drinking from the wine-and-cheese party early on till the wee hours.)

Whoever won got the reversi award. Tilbrook and Pike gave the first of many QED papers (this one called “the ultimate version”). Bill Joy then spoke about what was going on at Berkeley, and Wozniak about what was going on at BBN.

Most of the attendees knew what was transpiring at Berkeley, for Bill had distributed a four-page document, “Second Distribution of Berkeley Software for UNIX.” You could get a 2,400-foot 800BPI tape for \$60.

The tape contained the Pascal system, “a new version of the ‘ex’ editor,” “a new shell ‘csh,’” a new troff macro package (-me), and a new mail program.

I’ll let Dave Yost have the last word:

I witnessed BSD history being made quite casually at the Toronto USENIX that year. Bill Joy was standing in the lobby at the rear of the auditorium

when Jim Kulp from IIASA (in Laxenburg, Austria) approached him to pitch “Job Control.” He had it all working and wanted to offer it to Berkeley for inclusion in the distribution. I was excited; it sounded great. Bill, whose job it was to receive such submissions and then do a lot of the work integrating them, was noncommittal but said he’d look at it. The rest is history.

Postscript

A Note from a Veteran

Hi,

I’m one of Brian [Hudson]’s “15-year-olds” (“20 Years Ago in U[SE]NIX,” *login*, April 1999), though now of course I’m 35. I was quite pleased to see Brian and those of us lucky enough to be his students at the time mentioned in your column. Brian taught us enough about C and about UNIX internals and system administration so that we students could run and enhance the system on our own.

This system (an 11/70) supported activities such as class scheduling and student records, including our grades, and our school newspaper was laid out using our Diablo daisy-wheel printer. This in addition to our favorite activities: extending the system, writing and enjoying games, etc.

We had an awful lot of fun back then. We learned a lot of good and useful skills. We had the pleasure of seeing the results of

USENIX BOARD OF DIRECTORS

Communicate directly with the USENIX Board of Directors by writing to: <board@usenix.org>.

President:

Andrew Hume <andrew@usenix.org>

Vice President:

Greg Rose <ggr@usenix.org>

Secretary:

Peter Honeyman <honey@usenix.org>

Treasurer:

Dan Geer <geer@usenix.org>

Directors:

Jon “maddog” Hall <maddog@usenix.org>

Pat Parseghian <pep@usenix.org>

Hal Pomeranz <hal@usenix.org>

Elizabeth Zwicky <zwicky@usenix.org>

Executive Director:

Ellie Young <ellie@usenix.org>

CONFERENCES

Judith F. DesHarnais

Registration/Logistics

Telephone: 714 588 8649

FAX: 714 588 9706

Email: <conference@usenix.org>

Cynthia Deno

Marketing/Exhibitions/Publicity

Telephone: 408 335 9445

FAX: 408 335 5327

Email: <display@usenix.org>

Daniel V. Klein

Tutorials

Telephone: 412 421 2332

Email: <dvk@usenix.org>

that work, and the trust and feeling of community that developed.

Farrell Woods
<ftw@zk3.dec.com>

Call for Tutorial Proposals

by Daniel V. Klein

Tutorials Coordinator

<dvk@usenix.org>

In an effort to continue to provide the best possible tutorials to its membership, the USENIX Association is soliciting proposals for future tutorials. The tutorial proposals can cover any subject, ranging from reasonably introductory to advanced materials, although one should avoid overly generalized introductory materials (thus, a one-day tutorial on "Introduction to C Programming" is not the sort of thing we are usually looking for). Previous conferences have included tutorials on such diverse topics as UNIX Network Programming, High Availability, Topics in System Administration, Multi-Threaded Programming, UNIX Kernel Internals, Performance Tuning & Monitoring, Security, and Software Contracts & Intellectual Property, among many others.

In general, we like to categorize our tutorials as "introductory tutorials for advanced people," but some are "Advanced tutorials for advanced people." Tutorial instructors are remunerated for their presentations and have their conference registration and reasonable expenses paid for.

Tutorials usually run for a full day (six hours of class time plus morning, lunch, and afternoon breaks), although the smaller symposia and the LISA conference also hold half-day (three-hour) tutorials. Your proposal should include a statement of what you want to teach and a coherent outline of your tutorial – not simply a list of what you want to cover, but the order in which you want to cover it, with an estimate of the amount of time for each subject. We need to know that you can comfortably fill the time but not overfill it (i.e., that you won't discover at 4:30 that you have another three hours of slides left to present). Knowing in advance that you'll run until 6 p.m. is fine, so long as you warn your students ahead of time. Running until 7 p.m., though, almost guarantees that you will have unhappy students. If you have any supplementary materials to distribute (copies of papers, shell, Tcl, or Perl scripts, source code, illustrations, etc.), indicate the volume of supplementary material, along with a rough count of the number of slides you will be presenting

during class. (Historically, a typical tutorial takes between 75 and 200 slides, optionally with up to 200 pages of supplementary material). If possible, include a couple of sample slides (one with text, one with a graphic) with your proposal. If you have already written a complete or draft course, a copy of the current materials would be useful.

If you will be presenting or distributing any source code, we need to know whether it is copyrighted by someone other than you. If you do not hold the copyright, you must be able to demonstrate that you have permission to use this material (we want to avoid requiring course attendees to have a source license). Because the USENIX tutorials fall outside of the "fair use" clause of the U.S. copyright code, the same rules apply for supplementary papers or reports.

Finally, your proposal should also include a summary of your previous teaching or lecturing experience, as well as a couple of references (that is, one or two people who have seen you teach that we can contact). These may be your students, supervisors, or colleagues.

Remember, this is just a proposal, so nothing you submit will cast in concrete. You may later decide to change some ordering of materials, or we may suggest some changes. You needn't worry about getting it perfect the first time around. What we are trying to do is get a very

WEB SITE

<<http://www.usenix.org>>

MEMBERSHIP

Telephone: 510 528 8649
Email: <office@usenix.org>

PUBLICATIONS

Jane-Ellen Long
Telephone: 510 528 8649
Email: <jel@usenix.org>

USENIX SUPPORTING MEMBERS

C++ Users Journal
Cirrus Technologies
Cisco Systems, Inc.
CyberSource Corporation
Deer Run Associates
Greenberg News Networks
Hewlett-Packard India Software Operations
Internet Security Systems, Inc.
Microsoft Research
MKS, Inc.

Motorola Australia Software Centre
New Riders Press
Nimrod AS
O'Reilly & Associates
Performance Computing
Questra Consulting
Sendmail, Inc.
Server/Workstation Expert
TeamQuest Corporation
UUNET Technologies, Inc.
Windows NT Systems Magazine
WITSEC, Inc.

solid feel for what you are offering. You must sweat out some of the details, but needn't go too crazy over them.

All tutorial proposals are kept in mind when the tutorial program is chosen for a major USENIX conference or for one of our smaller workshops or symposia. If you feel that your proposal would be especially suited for a particular venue, please note that in your cover letter. Please send your proposals to dvk@usenix.org, or by physical mail to:

Daniel Klein, USENIX Tutorial Coordinator
5606 Northumberland ,
Pittsburgh, PA 15217-1238

Be sure to include both an electronic and a physical address and a phone number. All proposals will be acknowledged upon receipt.

New Releases of *BSD and Debian Linux OSes Given Away at USENIX Annual Conference

by **Cynthia Deno**

USENIX Marketing Director
cynthia@usenix.org

USENIX is providing grants to the OpenBSD, FreeBSD, NetBSD, and Debian Linux development projects, to support each of them in issuing new releases. USENIX has subsidized the production costs for releases of OpenBSD 2.5, FreeBSD 3.2, NetBSD 1.4, and Debian 2.2. The new releases will be available for distribution through each project's usual channels. As a bonus, copies of each of these new releases will be given to every technical-session registered attendee at the 1999 USENIX Annual Conference.

Support of the new releases continues USENIX's long-standing support of the

development process for open source software, helping to ensure that development will be characterized by intense yet healthy competition. The FREENIX track at the annual conference is another part of this effort. It is devoted to high-level technical discussion of open source software. FREENIX offers peer-refereed papers, expert talks, and evening sessions hosted by leading developers.

The 1999 Annual Conference takes place June 6-11, in Monterey, California. Programs for the tutorial and technical sessions, including the FREENIX track, and associated events are online. Please see <http://www.usenix.org/events/usenix99>.

USACO Wins Baltic Olympiad

by **Rob Kolstad**

Head Coach, USA Computing Olympics
kolstad@usenix.org

The U.S.A. team of six high school students won the informal team competition at the Baltic Olympiad in Informatics, an annual international programming contest held this year in Riga, Latvia, on the weekend of April 18, 1999. Six Baltic countries attended, with the U.S.A. invited as a guest country.

Each of two days of competition included a five-hour round of programming in which each student, working individually, was given three problems. At the end of each round, solutions were scored against sets of judges' test data.

Reid Barton, 15, of Boston, Massachusetts, won second place individual overall. Boulder, Colorado, resident Daniel Wright, 18, placed third overall. Other members of the winning U.S. team were Percy Liang, 16, Arizona; Po-Shen Loh, 16, Wisconsin; Jon McAlister, 17, Texas; and Kenn Hamm, 16, New York. The

team of six was chosen from the best performers in the country in a series of U.S.A. Computing Olympiad (USACO) programming contests held over the past six months. USACO coach Greg Galperin served as Team Leader.

Outside the contest, the trip gave students the opportunity to meet champion programmers from the other countries attending, as well as to tour the beautiful city of Riga, the capital of Latvia.

Here is a typical problem from the contest:

Given two sets A and B of strings, determine the shortest string which is a concatenation of strings from set A and which is also a concatenation of strings from set B. The sets A and B can have up to 100 strings of up to 50 characters each. Your program has five seconds to run on a Pentium-200.

USACO (<http://www.usaco.org>) is fully supported by USENIX.

Board Meeting Summary

by **Ellie Young**

Executive Director

<ellie@usenix.org>

Here is a summary of the actions taken at the regular meeting of the USENIX Board of Directors held on February 21, 1999, in New Orleans, Louisiana.

Attendance: USENIX Board: Rose, Geer, Hall, Honeyman, Zwicky, Pomeranz. USENIX Staff: Young, Berkowitz, Deno, DesHarnais. SAGE Executive Committee members: Wilson, Miller, Gittler, Gassaway, Dijker.

Budget 1999

The 1999 revised Budget was presented, reflecting the actions of the Board at its November meeting, the SAGE Executive Committee changes, and other adjustments. The Board voted to adopt the 1999 Budget.

Proposals for Funding Good Works

The Board agreed to develop a policy describing the types of activities and projects that should be funded under our Good Works program.

Student Programming Contest: A proposal from Evi Nemeth of the University of Colorado for \$1000 to sponsor their team to attend the annual ACM programming contest in Eindhoven was approved.

Polytechnic University/UNH Proposal: A proposal to continue funding the Polytechnic University's efforts with the United Neighborhood Houses of New York for \$65,000 was approved. In the coming year, the project may expand to several neighborhoods in the Boston area.

Future City Competition: A proposal from Polytechnic University for \$36,000 to sponsor and host the Future City Competition was approved. The competition attracts under-served students, a high proportion of whom are female, and it is based on the *SIM City* software program.

OpenBSD Sponsorship: A proposal from Pomeranz that USENIX sponsor the OpenBSD 2.5 release was approved. This will allow the project to master, assemble, and shrinkwrap the next release run, and CD sets will be distributed free of charge to USENIX conference attendees at the annual USENIX technical conference in Monterey (see p. 82 for more news). It was also agreed that we would welcome similar proposals from other projects that need assistance.

Investment Policy

A new guidelines and investment policy, which was drafted by a committee, was approved by the Board and will be included in the USENIX Association Policy Document.

USENIX Standards Activities

The Board reviewed the report from Stoughton's standards activities for January through March. It was agreed to

supply extra funds if appropriate representatives from the Linux community can be found to attend the meetings.

OSDI 2000

It was agreed that Mike Jones and Frans Kaashoek will serve as co-chairs and Honeyman will serve as board liaison.

USENIX 2000 Technical Conference

The proposal from Christopher Small to serve as program chair was accepted.

Extreme Linux Workshop

It was formally agreed that we would sponsor the second Extreme Linux Workshop at the Monterey conference.

Atlanta Linux Showcase

It was agreed that we should move forward on discussions with the Atlanta Linux Showcase representatives, including a plan to sponsor the first conference this fall and proceed with a second one. The staff was given funds in order to come up with a budget and proposal to supply logistical and other support for the ALS '99 conference. (See p. 78 for an announcement.)

Next Meeting

Will take place at the USENIX Conference in Monterey, California, on June 7. The Annual Meeting with the Membership will also take place there, on June 8.

3rd USENIX Windows NT Symposium

Monday-Thursday, July 12-15, 1999
Westin Hotel, Seattle, Washington, USA

Technical Program *Monday, July 12, 1999*

Keynote Address

Jim Cannavino
CEO/Chairman, *CyberSafe Corporation*

Business Computing: The Evolution of Opportunity

Technology in the business process has evolved significantly in the last 30 years: the 1960's-70's automated the back office, the 1980's-90's automated the front office. Now the Internet is automating the last inefficient piece of the business process: the consumer. Against this backdrop, Jim will look at today's Web-centric communication and its impact on security, as well as the advent of e-business and what it means for the future.

Jim Cannavino is CEO at CyberSafe, Inc. A former executive at both Perot Systems and IBM, he leads the company that develops leading security products for critical enterprise applications. In two years at Perot Systems, he grew the company from \$300 million to \$800 million. He held many positions during his 32-year tenure at IBM, retiring from the company as senior vice president for strategy and development. Prior to that, he led the company's restructuring of the PC business to form the IBM PC Company. Additionally, he forged IBM's alliance with Apple Computer and Motorola that led to the Power PC chip.

Cluster Computing

Session Chair: Werner Vogels, *Cornell University*

Efficient User-Level Thread Migration and Checkpointing on Windows NT Clusters

Hazim Abdel-Shafi, Evan Speight, and John K. Bennett, *Rice University*

High-End Workstation Compute Farms Using Windows NT

Srinivas Nimmagadda, Joshua LeVasseur, and Rumi Zahir, *Intel Corporation*

High-Performance Distributed Objects over System Area Networks

Alessandro Forin, Galen Hunt, *Microsoft Research, Microsoft Corporation*; Li Li, *Cornell University*; and Yi-Min Wang, *Microsoft Research, Microsoft Corporation*

Porting

Session Chair: Stephen Walli, *Softway Systems, Inc.*

MTEX - A Bridge for Migrating CAD Design Environment from UNIX to NT

Ty Tang, Vipul Lal, and Shesha Krishnapura, *Intel Corporation*

Porting Legacy Engineering Applications onto Distributed NT Systems

Nick Allsopp, Tim Cooper, P. Ftakas, *Parallel Applications Center*; and Patrick Macey, *SER Systems, Ltd.*

Porting a User-Level Communication Architecture to NT: Experiences and Performance

Yuqun Chen, Stefanos N. Damianakis, Sanjeev Kumar, Xiang Yu, and Kai Li, *Princeton University*

High Performance Systems

Session Chair: Jim Gray, *Microsoft Research, Microsoft Corporation*

Windows NT in a ccNUMA System

Bishop Brock, Gary Carpenter, Eli Chiprout, Mark Dean, Elmootazbellah Elnozahy, David Glasco, James Peterson, Ramakrishnan Rajamony, Freeman Rawson, Ron Rockhold, and Andrew Zimmerman, *IBM, Austin Research Lab*

The Record-Breaking Terabyte Sort on a 72-node Compaq Cluster

Pankaj Mehra and Samuel A. Fineberg, *Compaq Computer Corporation—Tandem Labs*

Millennium Sort: A Cluster-Based Application for Windows NT Using DCOM, River Primitives and the Virtual Interface Architecture

Philip Buonadonna, Joshua Coates, Spencer Low, and David E. Culler, *University of California, Berkeley*

Poster Session, Demonstrations, and Reception

Session Chair: Richard Oehler, *IBM T.J. Watson Research Center*

Poster and demo sessions will provide an open forum for symposium participants to describe their work in an informal setting. Anyone interested in setting up a poster or demo should send email to usenix-nt-posters@usenix.org.

Visit our website: <http://www.usenix.org/events/usenix-nt99>

3rd USENIX Windows NT Symposium

Monday-Thursday, July 12-15, 1999
Westin Hotel, Seattle, Washington, USA

Technical Program *Tuesday, July 13, 1999*

Keynote Address

Mendel Rosenblum, *VMware, Inc.*

VMware Virtual Platform Technology

VMware Virtual Platform is a software system that allows multiple operating system environments to run concurrently on a standard x86-based PC. By adapting some new twists to virtual machine monitor technology originally employed in the 1960's, the Virtual Platform provides virtualization of the non-virtualizable Intel x86 processor. VMware Virtual Platform also handles the large diversity of hardware available for the PC. The resulting system features both high performance and high portability, as well as ease of installation.

This talk will cover some of the major challenges of implementing in software a virtual machine monitor for a commodity, x86-based PC. The talk will also describe the solutions to these problems as implemented in VMware Virtual Platform.

Mendel Rosenblum, Ph.D., is Co-founder and Chief Scientist of VMware, Inc. He is a 1992 recipient of the National Science Foundation's National Young Investigator award and a 1994 recipient of an Alfred P. Sloan Foundation Research Fellowship. He was a co-winner of the 1992 ACM Doctoral Dissertation Award for his work on log-structured file systems. Dr. Rosenblum is an Associate Professor of Computer Science at Stanford University, where he leads the operating systems research group of the FLASH project. Together with his students, he developed the Hive scalable operating system, the SimOS complete machine simulator environment and the Disco scalable virtual machine monitor.

Real Time and Not

Session Chair: Susan Owicki, *InterTrust Technologies Corporation*

CPU Reservations and Time Constraints: Implementation Experience on Windows NT

Michael B. Jones, *Microsoft Research, Microsoft Corporation*; and John Regehr, *University of Virginia*

Hard Real-time with RTX on Windows NT

Mike Cherepov and Chris Jones, *VentureCom, Inc.*

Higher-Order Concurrent Win32 Programming

Riccardo Pucella, *Bell Laboratories, Lucent Technologies*

Indirection

Session Chair: Michael B. Jones, *Microsoft Research, Microsoft Corporation*

FIFS: A Framework for Implementing User-Mode File Systems in Windows NT

Danilo Almeida, *Massachusetts Institute of Technology*

Detours: Binary Interception of Win32 Functions

Galen Hunt and Doug Brubacher, *Microsoft Research, Microsoft Corporation*

Evaluating Windows NT TSE Performance

Alexander Ya-li Wong and Margo Seltzer, *Harvard University*

Internet

Session Chair: Karin Petersen, *Xerox Palo Alto Research Center*

A Case for a New CIFS Benchmark

Swami Ramany, *Network Appliance, Inc.*

HACC: An Architecture for Cluster-Based Web Servers

Xiaolan Zhang, Michael Barrientos, J. Bradley Chen, and Margo Seltzer, *Harvard University*

A Technique for Reducing Startup Latency in Mobile and Desktop Applications

Dennis Lee, Jean-Loup Baer, Brian Bershad, and Tom Anderson, *University of Washington*

NT Futures

George Spix, Chief Architect, Consumer Platforms Division, *Microsoft Corporation*, and Filipe Cabrera, Windows 2000 Storage Architect, *Microsoft Corporation*

In this session two of the most influential architects of Windows 2000 will talk about issues such as 64-bit, SMP, and cluster scaleup issues, the improved manageability of the data center product, and other interesting future developments. The session has a very informal nature with lots of room for discussion with the symposium participants.

Register now. On-line registration: <http://www.usenix.org/events/usenix-nt99>

LISA-NT—2nd Large Installation System Administration of Windows NT Conference

Wednesday-Saturday, July 14-17, 1999 • Westin Hotel, Seattle, Washington, USA

Technical Program *Friday, July 16, 1999*

Keynote Address

David P. Rodgers, Vice President, NT Program Office, *Compaq Computer Corporation*

More Than the Sum of the Parts: Combining Windows NT and Legacy Platforms

Rather than replace legacy platforms with Windows NT, organizations should combine the two platforms. Windows NT can supply flexibility, distributed computing, and Web capabilities, while legacy systems can compensate for NT's weaknesses in areas such as scalability, availability, and manageability.

David Rodgers currently oversees Compaq's effort to accelerate the adoption of Windows NT on Compaq hardware for mission-critical distributed transaction processing applications. Previously Vice President of Corporate Architecture at Sequent, he was also responsible for developing their Balance and Symmetry multiprocessor systems and the Dynix OS. During his ten-year stay at Digital Equipment Corporation, Rodgers headed the CPU development team on the VAX-11/780 super-minicomputer at Digital and was one of the architects of the Digital VAX computer family.

Large Installation Management

Session Chair: Aeleen Frisch, *Exponential Consulting*

Scalable, Remote Administration of Windows NT

Michail Gomberg, Craig Stacey, and Janet Sayrem, *Argonne National Laboratory*

A Network Machine Management System

Dave Roth, *Roth Consulting*

State-Driven Software Installation for Windows NT

Martin Sjolín, *Warburg Dillon Read*

Invited Talk

Tales from the Front—A Report from the Windows 2000 Beta Team

William Gloyeske, Team Manager for the Windows 2000 Beta Team, *Microsoft Research, Microsoft Corporation*

Session Chair: Matthew Olguin, *SEMATECH*

Non-Traditional Solutions

Session Chair: Ian Alderman, *Cornell University*

NFS and SMB Data Sharing Within a Heterogeneous Environment: A Real World Study

Alan Epps, Dr. Glenn Bailey, and Douglas Glatz, *Tektronix, Inc.*

Administering a Windows NT Domain Using a Non-Windows NT Primary Domain Controller

Gerald Carter, *Auburn University*

Radio Dial-in Connectivity to NT Networks

Kenneth May, *IBM Global Services*

"Again this year, I learned quality technical information—not marketing fluff—that will save my department many man-hours in the coming year."

—Todd Williams, *MacNeal-Schwendler Corporation*

"It is refreshing to have a group like this to share common experiences and problems. The depth of the knowledge in this group is unparalleled."

—Mike Kotnour, *GE Medical Systems*

Visit our website: <http://www.usenix.org/events/lisa-nt99>

LISA-NT—2nd Large Installation System Administration of Windows NT Conference

Wednesday-Saturday, July 14-17, 1999 • Westin Hotel, Seattle, Washington, USA

Technical Program *Saturday, July 17, 1999*

Invited Talk

Inside the Microsoft Network (MSN)

Chris Pinto, Director of Information Technology Group for MSN, *Microsoft Corporation*

Session Chair: Ralph Loura, *Cisco Systems, Inc.*

Windows NT Management Scenarios

Session Chair: John Holmwood, *TransCanada Pipeline Ltd.*

NT Security in an Open Academic Environment

Matthew Campbell, Andrea Chan, Robert Cowles, Gregg Daly, Ernest Denys, Patrick Hancox, William Johnson, David Leung, and Jeff Lwin, *Stanford Linear Accelerator Center*

Deployment of Microsoft Windows NT in a Design Engineering Environment

Jason Sampson, Elwood Coslett, Bob Paauwe, Russ Craft, Gary Washington, and Kevin Wheeler, *Intel Corporation*

NT Security Monitoring Using SNMP

Richard Reybok, *Lehman Brothers, Inc.*

Invited Talks

Securing Windows NT Network Services

Session Chair: Phil Cox, *Computer Incident Advisory Capability*

Securing Windows NT Services

David LeBlanc, *Microsoft Corporation*

Windows NT installs certain services by default, and others can be added either manually or as part of an application. The question then becomes "What happens when I turn a particular service off?" and "How does a particular service affect the network security of my machine?" This talk will help you to:

- Understand the services running on your machine
- Learn the security implications of each service.
- Understand how to write a secure service.
- Learn information on how to judge the security of a service from a vendor.

NT in the Firewall Environment

Elizabeth Zwicky, *Great Circle Associates*

As NT becomes a more and more important server platform, an increasing number of people need to run it in a firewall environment; people have NT bastion hosts, firewalls between cooperating NT machines, and NT firewalls. Unfortunately, solid information about NT in this environment is hard to come by, with both pro- and anti-NT camps producing more emotion than data about services, port numbers, and risks. This talk will attempt to provide some actual information about NT and firewalls.

Works-in-Progress

Session Chair: Paul Anderson, *University of Edinburgh*

Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The USENIX audience provides valuable discussion and feedback. Short, pithy, and fun, Works-in-Progress Reports (WIPs) introduce interesting new or ongoing work. We are particularly interested in presentation of student work. Prospective speakers should send a short one- or two-paragraph report to lisant99wips@usenix.org.

Register now. On-line registration: <http://www.usenix.org/events/lisa-nt99>

USENIX Windows NT 1999
3rd USENIX Windows NT Symposium
&
LISA-NT—2nd Large Installation System Administration of
Windows NT Conference

Monday-Saturday, July 12-17, 1999 • Westin Hotel, Seattle, Washington, USA

Windows NT Tutorial Program *Wednesday - Thursday, July 14-15, 1999*

Wednesday, July 14, 1999

Full Day Tutorial Sessions :

Windows NT/2000 Kernel Debugging & Crash Dump Analysis

Steven McDowell, *NCR Corporation*

Windows NT and UNIX Integration: Problems and Solutions

Phil Cox, *Networking Technology Solutions*

Morning Tutorial Sessions

DHCP/DNS

Greg Kulosa, *GNAC, Inc.*

The COM(+) Programming Model

Don Box, *DevelopMentor*

Afternoon Tutorial Sessions

Configuring and Administering Samba Servers

Gerald Carter, *Auburn University*

DCOM for Systems Administrators

Nicholas Schriber, *Collective Technologies Inc.*

Thursday, July 15, 1999

Full Day Tutorial Sessions

Windows NT Internals

Jamie Hanrahan, *Kernel Mode Systems*

Windows NT Security: Advanced Topics

Phil Cox, *Networking Technology Solutions*

Learning Perl

Daniel Klein, *Consultant*

Windows NT Performance Monitoring, Benchmarking and Tuning

Mark T. Edmead, *Windows NT Consultant*

Visit our website: <http://www.usenix.org/events/nt99>

Call for Papers

3rd Annual Atlanta Linux Showcase

October 12-16, 1999 • Cobb Galleria, Atlanta, Georgia • <http://www.linuxshowcase.org>

The Atlanta Linux Enthusiasts, USENIX, and Linux International, are pleased to announce the 3rd Annual Atlanta Linux Showcase. The Atlanta Linux Showcase will be held at the Cobb Galleria in Atlanta with a USENIX Tutorial program on October 12-13, followed by the General Conference sessions and Exhibition on October 14-16.

Important Dates

Submission deadline: *July 1, 1999*

Notification to authors: *July 15, 1999*

Camera-ready papers due: *September 8, 1999*

Registration material available: *August, 1999*

Speaker travel arrangements available: *August, 1999*

Overview

The Linux community is expanding at an ever increasing pace. ALS is a forum that brings together both experts and peers in our field. ALS will feature three conference tracks over three days with 40 speakers discussing the technical and business issues concerning the Linux Operating System. We invite you to submit paper proposals to enhance the invited talks, tutorials, and Birds-of-a-Feather sessions.

Topics

ALS is seeking papers that demonstrate Tools, Tutorials, or Case Studies in the areas of:

- Kernel
- Program Development
- Networking
- Applications
- Business Solutions
- Usability
- Security
- Unusual Applications

What to Submit

Papers should contain 1500 to 5000 words. After acceptance, papers may be edited for clarity and temporal changes until September 8, 1999.

Accepted papers will be shepherded through an editorial review process by a member of the program committee.

Selected papers will be included in the Conference proceedings, and at least one author will present the paper at the Showcase. Paper presentations will have approximately one hour including Q&A. Conference proceedings containing all papers will be distributed to attendees and will also be available from USENIX once the conference ends. We also ask that, if possible, copies of presentation slides be made available for wider distribution.

Papers accompanied by non-disclosure agreement forms are not acceptable, and will be returned unread.

Financial Assistance

Financial assistance for travel and accommodations is available. ALS requests that if your employer or other sponsor can cover some or all of these expenses, they do so. All speakers will receive free admission to the Showcase and an invitation to the welcome dinner Wednesday evening.

How and Where to Send Submissions

Please email your submission to papers@linuxshowcase.org in one of the following formats:

- Plain text with no extra markup
- Postscript formatted for 8.5" x 11" page
- HTML
- Applix Words
- UUencoded MS-Word

A cover letter with the following required information in the format below must be included with all submissions:

Authors:	Names and affiliation of all authors
Contact:	Primary contact for the submission
Address:	Contact's full postal address
Phone:	Contact's telephone number
Fax:	Contact's fax number
Email:	Contact's e-mail address
URL:	For all speaker/authors (if available)
Title:	Title of the submission
Needs:	Audio-visual requirements for presentation
Resume:	An informal resume of previous talks given. Lack of experience will not disqualify a speaker.
Abstract:	A short summary of the paper (100-200 words). This may be the paper's abstract.

If you enclose files as an attachment to your submission, please use MIME encoding.

We will acknowledge receipt of a submission by email within one week.

Tutorials

On October 12-13, there will be full and half-day tutorials in all areas and levels of expertise of Linux. If you are interested in presenting a tutorial at the conference, contact the tutorial coordinators:

Daniel V. Klein/Paul Manno

Email: tutorials@linuxshowcase.org

Birds-of-a-Feather (BOF) Sessions

BOF sessions are very informal gatherings of attendees interested in a particular topic. BOFs will be held in the evenings and may be scheduled at the conference or in advance by sending email to Matt Dinkins at bofs@linuxshowcase.org.

Registration Information

Complete program and registration information will be available in August 1999 at this Web site. If you would like to be kept up to date on ALS Information, sign up for our email list by sending the message 'subscribe als-announce' to: majordomo@linuxshowcase.org

2000 USENIX Annual Technical Conference

<http://www.usenix.org/events/usenix2000>

June 18-23, 2000

San Diego Marriott Hotel & Marina, San Diego, California

Important Dates

Paper submission deadline: *November 29, 1999*
Notification to authors: *January 26, 2000*
Full papers due for editorial review: *March 28, 2000*
Camera-ready papers due: *April 25, 2000*

Conference Organizers

Program Chair:

Christopher Small, *Lucent Technologies—Bell Labs*

Program Committee:

Ken Arnold, *Sun Microsystems*
Aaron Brown, *University of California at Berkeley*
Pei Cao, *University of Wisconsin*
Fred Douglass, *AT&T Labs—Research*
Edward W. Felten, *Princeton University*
Eran Gabber, *Lucent Technologies—Bell Labs*
Greg Minshall, *Siara Systems*
Vern Paxson, *International Computer Science Institute*
Liuba Shriru, *Brandeis University*
Keith A. Smith, *Harvard University*
Mark Zbikowski, *Microsoft*

Invited Talks Coordinators:

John Heidemann, *USC/Information Sciences Institute*
John T. Kohl, *Rational Software*

Overview

USENIX, founded twenty-five years ago, is the Advanced Computing Systems Association. Over the past quarter-century, the USENIX Association's membership has grown from its original core of UNIX users to a broad community of developers, researchers, and users with interests ranging from embedded systems to Tcl/Tk, from object-oriented programming and operating systems to network administration, and from Internet technologies and electronic commerce to using, managing, and researching Windows NT. The USENIX 2000 Annual Technical Conference seeks to bring together this broad community under a single roof to share the results of their latest and best work, find points of common interest and perspective, and develop new ideas that cross and break boundaries.

The three-day technical session of the conference includes a track of refereed papers selected by the Program Committee; a track of Invited Talks by experts and leaders in the field; and FREENIX, a track of talks and paper presentation on freely available POSIX-based software and systems. Refereed papers are published in the Proceedings, which are provided to Technical Session attendees, along with materials from the Invited Talks and FREENIX presentations.

Three days of tutorials precede the technical sessions with practical tutorials on timely topics.

Refereed Papers

The 2000 USENIX Technical Conference seeks to bring together the work, and the members, of the groups that make up the USENIX community. To that end, the Program Committee is interested in receiving submissions on a broad range of topics, including (but not limited to):

- Operating system and application structures for modern, commodity hardware, including extensible, embedded, distributed, and object-oriented systems.
- The impact of commodity hardware and software on the development of software systems.
- How the growing ubiquity of the Internet affects, and is affected by, the technological developments in the areas of electronic commerce, security, and heterogeneous and mobile computing.
- ActiveX, Java, CORBA, and other technologies that support mobile and reusable software components.
- The future of Tcl/Tk, Perl, and other scripting and domain-specific languages.
- Connecting, managing, and maintaining geographically distributed, heterogeneous networks of computers.

As at all USENIX conferences, papers that analyze problem areas, draw important conclusions from practical experience, and make freely available the techniques and tools developed in the course of the work are especially welcome.

Cash prizes will be awarded to the best paper and the best paper by a student.

Submitting a Tutorial Program Proposal

On Sunday, Monday, and Tuesday, June 18-20, USENIX's well-respected tutorial program offers intensive, immediately practical tutorials on topics essential to the use, development, and administration of advanced computing systems. Skilled instructors, who are hands-on experts in their topic areas, present both introductory and advanced tutorials covering topics such as:

- High availability and quality of service
- Distributed, replicated, and web based systems
- System administration and security
- Embedded systems
- File systems and storage systems
- Interoperability of heterogeneous systems
- Operating systems (Linux, BSD*, NT, etc.)
- Application development (threads, Perl, etc.)
- Intrusion detection and prevention
- Internet security

- Mobile code and mobile computing
- New algorithms and applications
- Systems application configuration and maintenance
- Personal digital assistants
- Security and privacy
- Web-based technologies

To provide the best possible tutorial slate, USENIX continually solicits proposals for new tutorials. If you are interested in presenting a tutorial, contact:

Dan Klein, Tutorial Coordinator

Phone: 412.422.0285

Email: duk@usenix.org

Submitting an Invited Talk Proposal

These survey-style talks given by experts range over many interesting and timely topics. The Invited Talks track also may include panel presentations and selections from the best presentations at recent USENIX conferences.

The Invited Talks coordinators welcome suggestions for topics and request proposals for particular talks. In your proposal state the main focus, including a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit via email to usenix2000-it@usenix.org.

Work-in-Progress Reports

Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentation of student work. To request a WIP slot, send email to usenix2000-wips@usenix.org.

Birds-of-a-Feather Sessions (BOFs)

The always popular evening BOFs are very informal, attendee-organized gatherings of persons interested in a particular topic. BOFs may be scheduled at the conference or in advance by contacting the USENIX Conference Office at 949.588.8649 or via email to conference@usenix.org.

FREENIX Track

FREENIX is a special track within the USENIX Annual Technical Conference. USENIX encourages the exchange of information and technologies between the commercial UNIX products and the free software world as well as among the various free operating system alternatives. FREENIX is the showcase for the latest developments and interesting applications in freely redistributable software including FreeBSD, Linux, OpenBSD, GNU, Apache, Samba, etc. The FREENIX track will cover the full range of software which is freely redistributable in source code form, with pointers to where the code can be found.

Additional information on preferred topics and what to submit will be available in June 1999.

How to Submit a Paper to the Refereed Track

Authors are required to submit full, complete papers by Monday, November 29, 1999. No papers will be accepted after 5:00 PM, Eastern time, Friday, December 3.

All submissions for USENIX 2000 will be electronic, in PostScript or PDF. Please follow the instructions for on-line submission at: <http://www.usenix.org/events/usenix2000/cfp/submit.html>.

Authors will be notified of receipt of submission via e-mail. If you do not receive notification, contact:

usenix2000-chair@usenix.org.

Papers should be 8 to 12 single-spaced 8.5 x 11 inch pages (about 4000-6000 words), not counting figures and references. Papers longer than 14 pages and papers so short as to be considered extended abstracts will not be reviewed. More detailed author instructions will be available on the conference web site at: <http://www.usenix.org/events/usenix2000/cfp/guidelines.html>

It is imperative that you follow the instructions for submitting a quality paper. Specific questions about submissions may be sent to the program chair via email to:

usenix2000-chair@usenix.org.

A good paper will clearly demonstrate that the authors:

- are attacking a significant problem,
- are familiar with the literature,
- have devised an original or clever solution,
- if appropriate, have implemented the solution and characterized its performance using reasonable experimental techniques, and
- have drawn appropriate conclusions from their work.

Note: the USENIX Technical Conference, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers submitted to this conference that are under review elsewhere will not be reviewed. Papers accompanied by non-disclosure agreement forms can not be accepted, and will not be reviewed. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors will be notified by January 26, 2000. All accepted papers will be shepherded by a program committee member through an editorial review process prior to final acceptance for publication in the proceedings.

USENIX Exhibition

In the exhibition, the emphasis is on serious questions and feedback. Vendors will demonstrate the features and technical innovations which distinguish their products. For more information, including a current list of exhibitors, see <http://www.usenix.org/events/usenix2000/vendors.html>.

Contact:

Dana Geffner

USENIX Exhibition Coordinator

Phone: 831.457.8649

Email: dana@usenix.org

Program and Registration Information

Complete program and registration information will be available by March 2000 at the conference website: <http://www.usenix.org/events/usenix2000>. The information will be printable from a PDF file.

If you would like to receive the program booklet in print, please email your request, including your postal address, to conference@usenix.org.



MobiCom'99

The Fifth Annual International Conference on
Mobile Computing and Networking

August 15 - 20, 1999

Seattle, Washington, USA

<http://mobicom99.research.microsoft.com>

Sponsored by ACM SIGMOBILE

Technically Co-sponsored by the IEEE Communications Society
In cooperation with ACM SIGCOMM, SIGOPS, SIGMETRICS, and SIGMOD;
the USENIX Association; the IEICE; and the IEE

MobiCom'99 is the fifth of an annual series of international conferences dedicated to addressing the challenges of the wireless revolution. By bringing together researchers, practitioners, and visionaries from all over the world, MobiCom provides an environment where ideas flow freely and intellectual discussions happen easily between individuals instrumental in shaping the world of tomorrow.

Technical Sessions – Technical papers describing previously unpublished, original, completed research, were solicited on a wide variety of topics in mobile and wireless communications. This year, *the largest number of paper submissions were received*, which promises highly selective technical sessions covering the most up to date, ground-breaking work of today. A new **Next Century Challenges** session will be included. Papers in this session will challenge the mobile computing community with new technologies or visionary applications, and provide stimulating ideas or visions that promise to open up exciting avenues of mobile computing research.

Panels - The Future of Local Area Wireless Networking (Moderator: Marvin Theimer, Microsoft Research); Electronic Books (Moderator: Dan Russell, Xerox PARC) Global Satellite Communication Networks (Moderator: Satchandi Verma, Motorola); Living with Wearable Computers (Moderator: Margaret Orth, Media Lab, MIT)

Speakers - Speakers will include Dr. Rick Rashid, Vice-President of Microsoft Research, as the keynote address as well as Dr. Mark Weiser, Chief Technology Officer of Xerox PARC, as banquet speaker.

Co-located Workshops – Tackling the dominant issues of the day, these workshops will allow extended consideration of particular topics:

- Data Engineering for Wireless and Mobile Data (MobiDE'99)
- Discreet Algorithms and Methods for Mobile Computing and Communications (Dial-M'99)
- Wireless Mobile Multimedia (WoWMoM'99)
- Modeling and Simulation of Mobile and Wireless Systems (MSWS'99)

Tutorials – Several tutorials aimed towards advanced researchers, designers and practitioners of mobile computing will be given. Topics will cover subjects such as security, TCP for wireless networks, energy efficiency, channel coding basics, and simulation techniques for wireless networks.

Exhibition – The conference will feature an exhibition of the newest, cutting-edge offerings from a wealth of companies.

Location - The Bell Harbor International Conference Center is a global landmark, offering technologically advanced conference facilities, and is located right on the inner harbor of Seattle, Washington. The city is often referred to as "The Emerald City", and is considered one of the most livable cities in the world.



USENIX MEMBERS SAVE

DESIGN THE
COMPONENT-BASED
SYSTEM
THAT IS RIGHT
FOR YOU

**USENIX MEMBERS SAVE
\$500 on Cutter Consortium's Distributed
Computing Publications Package:**

Component Development Strategies. Find out how to use components and frameworks to build distributed applications successfully in this monthly newsletter.

The Corporate Use of Object Technology. Learn how companies are transitioning from procedural technologies to the use of object-based technology.

Componentware: Building It, Buying It, Selling It. Discover how different componentware initiatives can be managed to deliver coherent, integrated business solutions.

FIND OUT HOW you can develop successful component-based systems!

For more information, contact Megan Nields of Cutter Information Corp. at 781-641-5118 or mniels@cutter.com. Use priority code 130*6UX.
www.cutter.com/cds/

CUTTER
||| CONSORTIUM |||

TAKE
10%

**USENIX MEMBERS
TAKE 10% OFF** your
first-year subscription to any Cutter
newsletter:

- *Component Development Strategies*
- *Data Management Strategies*
- *IT Metrics Strategies*
- *Application Development Strategies*
- *Intelligent Software Strategies*
- *The Cutter IT Journal*

To get your 10% member discount, contact Megan Nields at 781-641-5118 or sign up at our Web site www.cutter.com and use priority code 0*6UX.
www.cutter.com

Publications Order Form**USENIX***Publications Order Form*

Quantity			Price: Member/List	Overseas Postage	Total
General Technical Conferences					
_____	1998 Annual	New Orleans, LA	\$32/40	18.	_____
_____	1997 Annual	Anaheim, CA	\$32/40	18.	_____
_____	1996 Annual	San Diego, CA	\$32/40	18.	_____
_____	1995 Annual	New Orleans, LA	\$30/39	20.	_____
_____	Summer 1994	Boston, MA	\$25/33	20.	_____
_____	Winter 1994	San Francisco, CA	\$30/39	20.	_____
_____	Summer 1993	Cincinnati, OH	\$25/33	18.	_____
_____	Winter 1993	San Diego, CA	\$33/40	25.	_____
_____	Summer 1992	San Antonio, TX	\$23/30	14.	_____
_____	Winter 1992	San Francisco, CA	\$30/39	22.	_____
Systems Administration (LISA and LISA-NT) Conferences					
_____	Systems Administration (LISA XII)	December 1998	\$32/40	12.	_____
_____	Large Installation System Administration of Windows NT	August 1998	\$18/24	12.	_____
_____	Systems Administration (LISA XI)	October 1997	\$30/38	12.	_____
_____	Systems Administration (LISA X)	September 1996	\$30/38	20.	_____
_____	Systems Administration (LISA IX)	September 1995	\$30/38	20.	_____
_____	Systems Administration (LISA VIII)	September 1994	\$22/29	20.	_____
_____	Systems Administration (LISA VII)	November 1993	\$25/33	18.	_____
_____	Systems Administration (LISA VI)	October 1992	\$23/30	25.	_____
Networking					
_____	Network Administration (NETA)	April 1999	\$18/24	11.	_____
_____	Intrusion Detection and Network Monitoring (ID)	April 1999	\$23/30	11.	_____
_____	High-Speed Networking	August 1994	\$15/20	9.	_____
Security					
_____	Security VII	January 1998	\$27/35	18.	_____
_____	Security VI	June 1996	\$27/35	18.	_____
_____	Security V	June 1995	\$27/35	18.	_____
_____	Security IV	October 1993	\$15/20	20.	_____
_____	Security III	September 1992	\$30/39	20.	_____
Operating Systems					
_____	Operating System Design & Implementation	February 1999	\$23/30	11.	_____
_____	Operating System Design & Implementation	October 1996	\$20/27	11.	_____
_____	Operating System Design & Implementation	November 1994	\$20/27	11.	_____
_____	Mach III Symposium	April 1993	\$30/39	18.	_____
_____	Microkernels & Other Kernel Architectures Symposium	September 1993	\$15/20	9.	_____
_____	Distributed & Multiprocessor Systems (SEDMS IV)	September 1993	\$24/32	14.	_____
Programming Languages					
_____	Conference on Domain-Specific Languages	October 1997	\$20/24	12.	_____
_____	6th Tcl/Tk Workshop	September 1998	\$22/28	12.	_____
_____	5th Tcl/Tk Workshop	July 1997	\$22/28	12.	_____
_____	4th Tcl/Tk Workshop	July 1996	\$22/28	12.	_____
_____	3rd Tcl/Tk Workshop	July 1995	\$29/34	20.	_____
_____	Conf. on Object-Oriented Technologies & Systems V	May 1999	\$24/32	12.	_____
_____	Conf. on Object-Oriented Technologies & Systems IV	April 1998	\$22/32	12.	_____
_____	Conf. on Object-Oriented Technologies & Systems III	June 1997	\$20/30	12.	_____
_____	Conf. on Object-Oriented Technologies & Systems II	June 1996	\$20/30	12.	_____
_____	Conf. on Object-Oriented Technologies & Systems I	June 1995	\$18/24	9.	_____
_____	Very High Level Languages	October 1994	\$23/30	10.	_____
_____	C++ Conference	April 1994	\$24/28	20.	_____

Quantity		Price: Member/List	Overseas Postage	Total
Other Symposia & Workshops				
_____	Workshop on Smartcard Technology	May 1999	\$22/28	12.
_____	2nd USENIX Windows NT Symposium	August 1998	\$18/24	12.
_____	USENIX Windows NT Workshop	August 1997	\$18/24	12.
_____	Symposium on Internet Technologies & Systems	December 1997	\$20/26	12.
_____	Electronic Commerce Workshop	August/Sept. 1998	\$20/26	12.
_____	Electronic Commerce Workshop	November 1996	\$20/26	11.
_____	Electronic Commerce Workshop	July 1995	\$20/26	10.
_____	Mobile and Location Independent Computing II	April 1995	\$18/24	9.
_____	Mobile and Location Independent Computing I	April 1995	\$15/20	8.
_____	UNIX Applications Development	April 1994	\$15/20	9.
_____	File Systems Workshop	May 1992	\$15/20	9.

Discounts are available for bulk orders of 10 or more of the same proceeding.

USENIX CD-ROM

_____	New Orleans '98	June 1998	\$65/90	3.50
_____	Anaheim '97	January 1997	\$65/90	3.50
_____	LISA X	September 1996	\$65/90	3.50

SAGE: Short Topics in System Administration Series

_____	#1: Job Descriptions for System Administrators, 2nd ed	\$5/7.50	3.50
_____	#2: A Guide to Developing Computing Policy Documents	\$5/7.50	3.50
_____	#3: System Security: A Management Perspective	\$5/7.50	3.50
_____	#4: Educating and Training System Administrators: A Survey	\$5/7.50	3.50

Note:

The member price also applies to members of EurOpen National Groups, AUUG, and JUS.

Reprints:

Reprints of individual papers from all proceedings are available for \$5.00 each. (To get a complete listing of conference papers, please refer to our Online Library Index, available through the WWW at <<http://www.usenix.org>>).

Payment Options

☐ Check enclosed payable to USENIX Association. ☐ Purchase order enclosed ☐ Visa ☐ MasterCard ☐ American Express

Account # _____ Expiration Date _____ Signature _____

Outside the USA? Please make your payment in US currency via one of the following:

☐ Check – issued by a local branch of a US bank ☐ Charge (Visa, MasterCard or equivalent) ☐ International postal money order

SHIP TO: _____

Shipping Information: Please allow 2-3 weeks for delivery. Shipping fees for domestic and Canadian orders are included. Please add postage fee for overseas orders (shipped via air printed matter).

Total Price of Publications _____

Calif. res. add sales tax: _____

Overseas Postage _____

TOTAL ENCLOSED \$ _____

If you are a member, please include your name and membership # to receive member price: _____

☐ If you are not a member and wish to receive our membership information packet, please check this box.

Please return this order form and your payment to:

USENIX Association

2560 Ninth Street, Suite 215

Berkeley, CA 94710.

Fax: 510/548-5738 Phone: 510/528-8649 <office@usenix.org> <<http://www.usenix.org>>

motd

by Rob Kolstad



Dr. Rob Kolstad works as program manager organizing computer security conferences. Longtime editor of *login*, he is also head coach of the USENIX-sponsored USA Computing Olympiad.

<kolstad@usenix.org>

Respect

You can well imagine that the tragedy in Littleton is big news in my Colorado neighborhood; it seems that everyone is trying to figure out how and why it happened. I fear that too often this is so that they can point a finger of blame and somehow reach a sort of personal "resolution" of the issue.

The local media are full of articles sharing "background information," "new facts," reports of psychologists, and a host of other random data that people might use to try to find a way to come to grips with the horrifying reality of over a dozen dead. The bizarre synchronicity of the event and debate over gun control in our state government compounds the tongue-wagging. People actually say things like, "If only teachers carried their own handguns . . ."

In their search for blame, others look to the parents. "How could they not have noticed the bomb-making and sawed-off shotgun barrels?" Indeed. Yesterday's rumor (surely to be resolved by the time this reaches print) suggested criminal charges were in order for the parents, the parents of perpetrators of ages 17 and 18. Even the law recognizes 18-year-olds as fully responsible for their actions.

My dental hygienist seemed to have a rational and prescriptive viewpoint. She suggested that the real problem is a lack of a certain kind of respect on the part of everyone. Obviously, the Littleton shooters did not respect their victims. The murderers would have us believe they were treated poorly by other groups of people at their school. While potentially a real problem, of course, this would never justify murder.

I visited Jet Propulsion Laboratories a week ago for a weekend high school competition revolving around designs of space settlements. One 16-year-old male contestant wore a rock group T-shirt whose four-inch-high words included the 12-letter obscenity that some believe is the most powerful of button-pushers. Four-inch-high red letters. Right there in front of God and everyone.

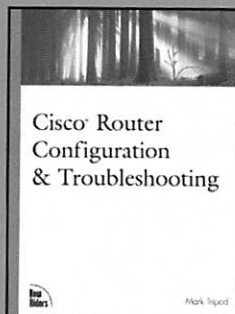
I asked one of the organizers if anyone was going to do anything. They suggested that he had already achieved his intended goal just by motivating me to ask the question. Furthermore, I was slightly castigated for suggesting that maybe his freedom of expression might offend some number of our gender-balanced group. I have not discussed the issue with him since the Littleton catastrophe.

But the juxtaposition of these two events has driven home the notion of "respect." The USENIX community is one of the most tolerant groups with which I have had the privilege of associating. Just check out the "colored dots" on the nametags of conference attendees to see what I mean. Every possible interest, lifestyle, fashion statement, and attitude can be found. The community seems to judge people far more on their technical achievements, diction, and attitude than those other things. The tolerance can even be thought of as respect in a certain technical context.

Maybe we need to share this attitude of respect with our friends, neighbors, and associates. I'm afraid I honestly don't know how to do it without being somewhat of a jerk. "Hey dude, your shirt is disrespectful of mothers, women in general, and males. Please remove it if you wish to stay at our contest." He probably wouldn't think better of me, adults, authority, or anything else. But I'm going to try. I hope you will, too.

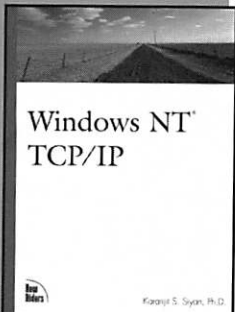
THE FUTURE IS IN THE HANDS OF THOSE PREPARING FOR IT.

ISBN: 0-7357-0024-9



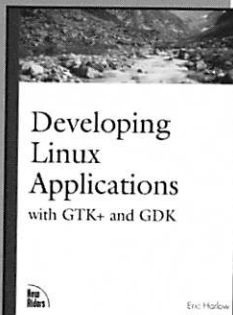
Prepare for professional success using resources from **New Riders, Cisco Press and MTP**, now available for **20% less** for **USENIX/SAGE** members only!

ISBN: 1-56205-887-8

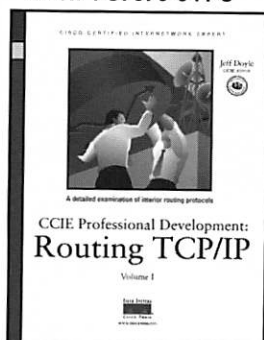


Call 1-800-428-5331 and mention source code **USNX** to get best-selling books from authors who are influencing tomorrow's technology today.

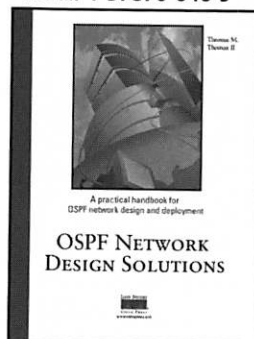
ISBN: 0-7357-0021-4



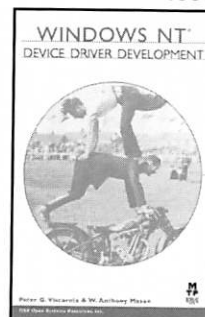
ISBN: 1-57870-041-8



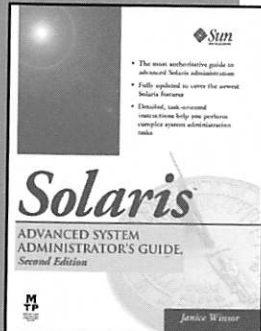
ISBN: 1-57870-046-9



ISBN: 1-57870-058-2



ISBN: 1-57870-039-6



Call 1-800-428-5331
mention source code **USNX** and get 20% off!

**New
Riders**

CISCO SYSTEMS

CISCO PRESS

**M
TP**
MACMILLAN
TECHNICAL
PUBLISHING
U.S.A.

www.newriders.com • www.ciscopress.com • www.macmillantech.com

Shipping and handling additional. MasterCard/VISA/American Express accepted.

Copyright ©1999 Pearson Education

CONNECT WITH USENIX



MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>



WEB SITE

<http://www.usenix.org>



EMAIL

login@usenix.org



COMMENTS? SUGGESTIONS?

send email to jel@usenix.org



CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are August 3, 1999, and October 5, 1999.

USENIX

The Advanced Computing Systems Association

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

136935